Variable Fractional Digital Delay Filter on Reconfigurable Hardware

A Thesis

Submitted to the Faculty

of

Drexel University

by

Karthik Ramu Sangaiah

in partial fulfillment of the

requirements for the degree

of

Master of Science in Computer Engineering

June 2012

Dedications

I dedicate this thesis to my father, who taught me the importance of taking on new challenges and to keep an open mind. I also dedicate this thesis to my mother and sister. Without their support and encouragement, I would not be able to be where I am today.

Acknowledgments

I would like to express my appreciation to my advisor Dr. Prawat Nagvajara for his understanding and support through this project. His advice and ideas were vital to the design and testing stages of this project.

I also would like to thank Dr. Baris Taskin and Dr. Nagarajan Kandasamy for being thesis committee members.

Additionally, I would like to thank Kevin Cunningham, Andrew Cebulski, Mike Lui, Mikhail Chernyavskiy, and Eugin Cherkansky for their advice during this project.

Table of Contents

List of Tables

List of Figures

Abstract

Variable Fractional Digital Delay Filter on Reconfigurable Hardware
Karthik Ramu Sangaiah
Prawat Nagvajara, Ph.D.

This thesis describes a design for a variable fractional delay (VFD) finite impulse reponse (FIR) filter implemented on reconfigurable hardware. Fractionally delayed signals are required for several audio-based applications, including echo cancellation and musical signal analysis. Traditionally, VFD FIR filters have been implemented using a fixed structure in software based upon the order of the filter. This fixed structure restricts the range of valid fractional delay values permitted by the filter. This proposed design implements an order-scalable FIR filter, permitting fractionally delayed signals of widely varying integer sizes. Furthermore, the proposed design of this thesis builds upon the traditional Lagrange interpolator FIR filter using either a software-based coefficient computational unit or hardware-based coefficient computational unit in reconfigurable hardware for updating the FIR coefficients in real-time. Traditional Lagrange interpolator FIR filters have only permitted fixed fractional delay. However, by leveraging today's (2012) low-cost high performance reconfigurable hardware, an FIR-based fractional delay filter was created to permit varying fractional delay. A software/hardware hybrid VFD filter was prototyped using the Xilinx System Generator toolkit. The resulting real-time VFD FIR filter was tested using System Generator, as well as Xilinx ISE and ModelSim.

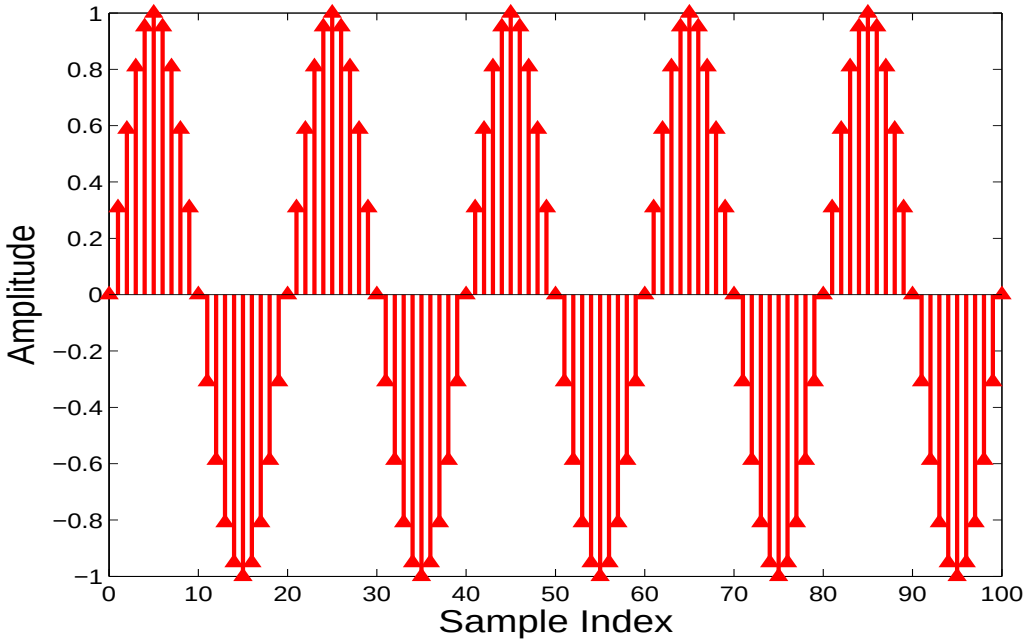
Chapter 1: Introduction

In many modern digital signal processing applications, a fractional delay (FD) of a signal is required as opposed to a unit delay. Such applications include echo cancellation, modeling human voice pitch, musical signal analysis, sampling rate conversion, and timing synchronization [1]. Several fractional delay implementation methods are described in detail in [1]. A common requirement in most of these applications is that a real-time adjustable fractional delay value must be updated as well. Typically, these applications require the use of a variable fractional delay (VFD) filter. A typical VFD filter will take a sampled signal input and output re-sampled fractionally delayed signal.

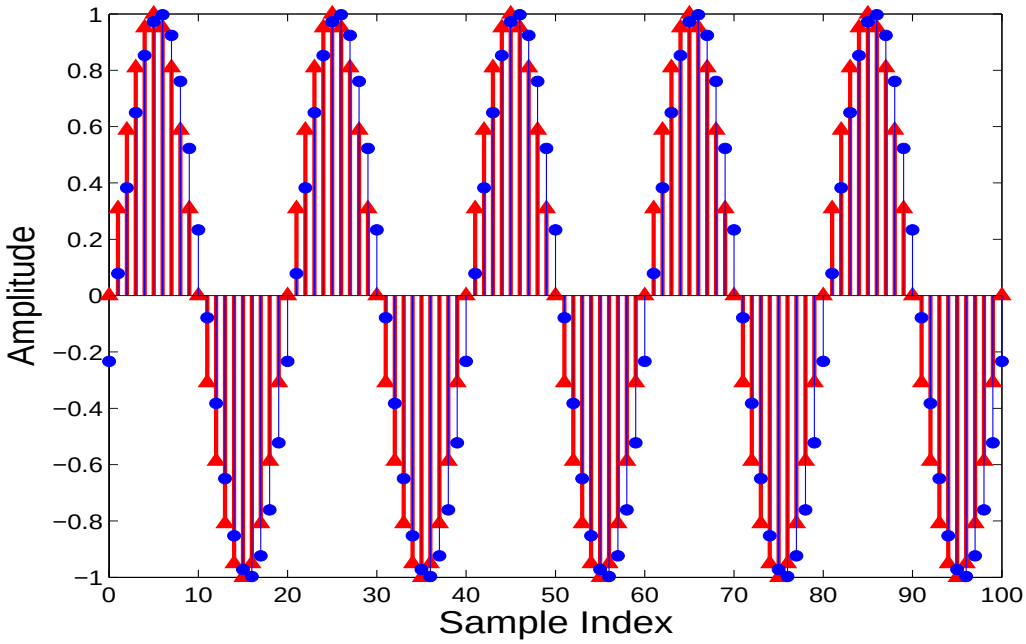
The main purpose of this work is to create a new hardware-based filter that can delay a signal with an arbitrary sized delay between sample points. The previously explored implementations of this application have been able to delay signals with non-variable and variable fractional delay as a software application on a commodity DSP processor. However, there are performance limitations with a software implementation that can be exploited by catering this application for a hardware-based solution. This work highlights typical designs and algorithms that have been implemented in previous research and follows up these designs with Field-Programmable Gate Array (FPGA)-based solutions that exploit the limitations of the software implementations.

For a basic understanding of a FD filter, Figure 1.1 graphically illustrates the

desired operation with a fixed delay parameter.



(a)



(b)

Figure 1.1: Graphs of (a) Sampled Input Sine Wave and (b) Fractionally Delayed Input Sine Wave

As shown in Figure 1.1, the main premise for this application is to interpolate the values between sample points to achieve a recreated delayed signal. Figure 1.1a illustrates a basic sampled sine wave based upon a sample index. Figure 1.1b depicts the same input signal in red and a fractionally delayed version of the input signal with a fractional delay of .33 samples. Using this very rudimentary example, it is visually evident of the desired delayed signal at each of the sampling points.

The fundamental problem to solve in designing VFD filters is based upon manipulating available sampled data to interpolate the desired delayed signal while minimizing the error of the approximated and actual delayed signal. The design of algorithms for variable and non-variable fractional delay filters have been discussed over the last 30 years [**?**]. Portions of the mathematics and systems theory behind these designs, such as Lagrange interpolation, have been prevalent since the 18th century. The implementations of these mathematical algorithms have been presented in several papers across the engineering community. The Farrow structure, the maximally-flat Lagrange interpolator, and other IIR VFD filters have been explored from a theoretical design point [1], [2], [3]. In practice, these filters are often implemented with commodity DSP processors. These DSP-targeted implementations are limited by the system specifications, such as sampling rate, output flow rate, and bandwidth, as well as the DSP hardware bottlenecks, such as bus performance and a limited number of computational units.

As a result, the same arithmetically intense operations executed in these software algorithms can be executed much faster and scaled easier in a hardware-based design. Over the last three years, groups, such as Ramirez-Conejo et al., have started to

implement these theoretical designs into reconfigurable hardware, specifically FPGAs [4], [5].

This work presents a comparison of known fractional delay filter implementations, and proposes a design for VFD filters on reconfigurable hardware that takes advantage of the hardware platform. The proposed designs is a real-time order-scalable Lagrange interpolator-based VFD FIR filter. The targeted platforms for this design are the Xilinx Spartan-6 FPGA and Xilinx Virtex-6 FPGA.

1.1 Past Research

The research in this field has been explored in theoretical design, software implementations, and hardware implementations.

Laakso, Valimaki, Cain, and Yardim detail digital FIR filter software implementations of FD filters in approximating an ideal FD signal [6],[7],[8], [9]. Laakso, Cain, and Yardim detail a windowed *Sinc* function approximation [7],[8], [9] . Laakso and Valimaki consider using a lowpass FD filter design using a low-order spline function [6]. Laakso and Valimaki also details designing a maximally-flat FIR approximation using Lagrange interpolation, a weighted least-squares (WLS) approach to FD signal approximation, and a quasi-equiripple FD signal approximation using Oetken's method [6]. Reviewed in [10] and [2], Kootsookos et al. derived that based upon the design choices, these FIR filter-based FD approximations, windowed *Sinc* function, Lagrange interpolation, and the maximally-flat error approximation are all equivalent.

The field of VFD filters has been well researched with the common base structure of the Farrow structure [**?**]. In many applications, the VFD filter is typically based

on a variant of the Farrow structure, which allows for an adjustable delay parameter that can be updated real-time; however, other VFD filter types have been explored in academic papers using IIR-based filters [11], [12], improved weighted least-squares FIR filter design [13], and minimax-based FIR filters[14].

Laakso and Valimaki also discuss the design of IIR approximation FD filters as all-pass filters [1], [6]. These techniques include least squares phase approximation, least squares phase delay approximation, and maximally-flat group delay approximation using a Thiran allpass filter.

Rameriz-Conejo et al. and Nithirochananont et al. detail current implementation techniques of FD filters on FPGA hardware [4] and [5]. Rameirez-Conejo et al. proposed a time-domain wideband fractional delay filter using a multi-rate Farrow structure design [4] . This filter design uses a Lagrange interpolator to up-sample the input signal in the first stage of the system to reduce the required bandwidth. The main filter stage is implemented using a distributed arithmetic (DA) design to reduce the number of embedded multipliers necessary from the FPGA. This design also targets using general purpose multipliers. Thus, the filter was designed with DA to improve performance and reduce the necessary hardware. Also, this system incorporated a LUT architecture to store pre-computed coefficients of the filter. Nithirochananont et al. detailed the design of a VFD filter using a Taylor series structure to reduce the number of arithmetic operations required compared to a Farrow structureReference [5]. This FPGA implementation used the parallel distributed arithmetic method, which improves upon the traditional DA method by ways of increasing speed of computation. A LUT architecture was also implemented to store the pre-computed

coefficients of the filter, as well as a LUT for potential fractional delay values.

1.2 Contributions of the Thesis

This thesis reports on two implementation paradigms of a proposed fractional delay filter architecture that is designed with the target platform of custom hardware. To make use of the design flexibility and to reduce cost, the proposed fractional delay filters can also be implemented on FPGA. The proposed design builds upon previous research of the maximally-flat Lagrange interpolators and the Farrow structure and is designed to permit widely varying fractional delay applications. The proposed filter design is designed to be order-scalable, which will permit widely varying fractional delay. The design decisions for this filter will be further explained in Chapter 3.

Two implementations have been designed and functionally verified for the VFD filter architecture. As illustrated in Figure 1.2, the high-level layouts outline a software/hardware hybrid VFD filter and a fully hardware computation VFD filter.

The proposed implementation of the VFD FIR filter for widely varying fractional delay is presented. The design includes a fixed maximal size FIR filter with the coefficients corresponding to a Lagrange interpolator. A Lagrange coefficient computational unit computes the Lagrange FIR coefficients in real-time based upon the desired order of the filter and fractional delay. The software/hardware hybrid VFD filter includes a software Lagrange coefficient computation unit that sends the respective coefficients into the hardware FIR filter within the FPGA. The fully hardware design includes a custom hardware Lagrange coefficient computation unit that recomputes the coefficients output to the respective FIR filter coefficients within the

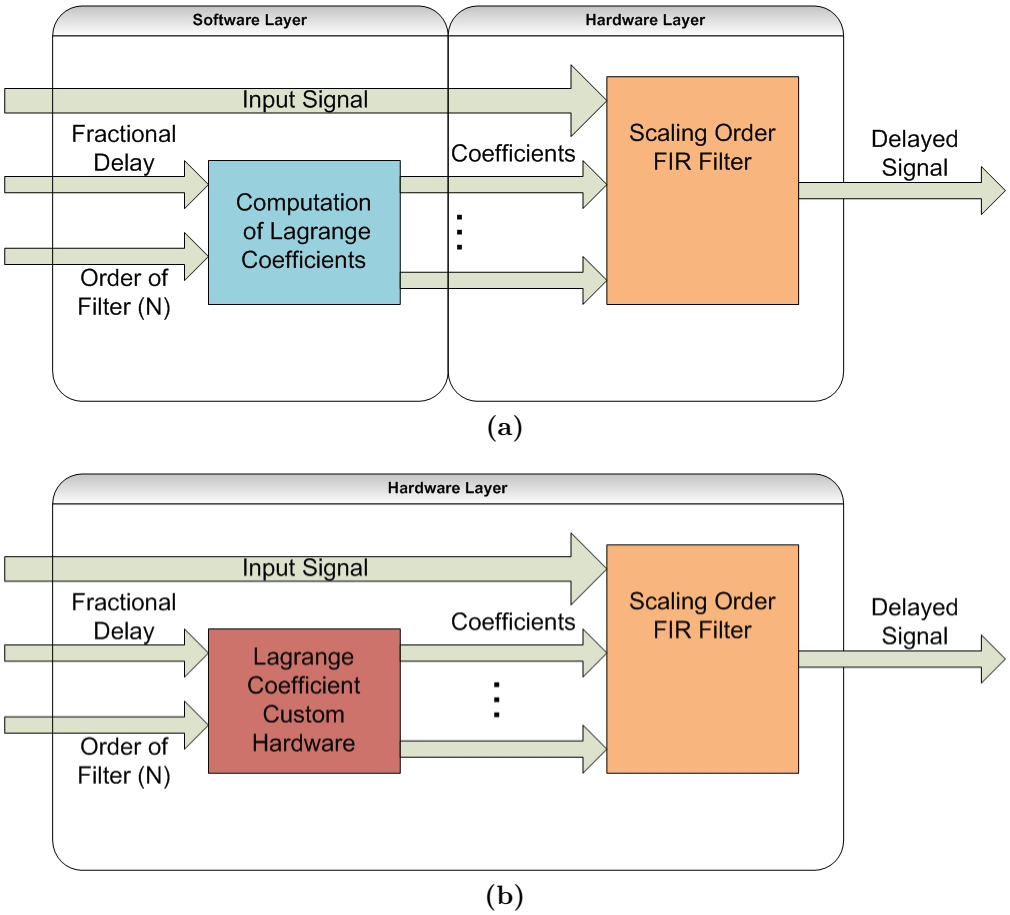


Figure 1.2: (a) Software/Hardware Hybrid VFD FIR Filter and (b) Hardware VFD FIR Filter

FPGA.

These two implementations of the proposed VFD filter design were designed and tested using the Mathworks Simulink environment, specifically using the Xilinx System Generator toolkit for FPGA development. A prototype of software/hardware hybrid filter implemented on the Xilinx ML605 Virtex-6 FPGA development board. This prototype was implemented, tested, and functionally verified using the System Generator toolkit. The second implementation of the filter design has been designed,

with each internal component tested in ModelSim and synthesized using the Xilinx ISE software suite.

1.3 Thesis Overview

Chapter 2 introduces the basic principles and key concepts of ideal and practical fractional delay filters and interpolator models. Chapter 2 also provides a overview of the common VFD algorithms.

Chapter 3 discusses the designs of the proposed and prototyped VFD filter architecture. Chapter 4 provides the performance analysis, functional verification, and requirement criteria of the implementations.

Finally, Chapter 5 presents the conclusion of the thesis, summarizing the proposed VFD filter design findings and potential future work.

Chapter 2: Overview of Fractional Delay Concepts and Filter Models

This chapter will review the basic principles of designing fractional delay filters. First, the ideal model of FD filters is presented from [15]. The ideal delay model is based on Shannon's signal reconstruction formula of Shannon's theorem. Following the discussion on the ideal model of FD filters, design methods for implementable digital filters approximating fractionally delayed signals is presented. As mentioned in Chapter 1, many design methods have been discussed theoretically and as software techniques. This chapter discusses useful techniques that are implementable and build the foundation of the proposed design.

2.1 Ideal Model of Fractional Delay Filters

2.1.1 Continuous-Time of Delay Systems

Prior to addressing fractional delay in discrete time, the concept of fractional delay in continuous-time is discussed. A linear system that delays an input continuous-time signal $\boldsymbol{x_c(t)}$ by $\boldsymbol{\tau}$ will output a continuous-time delayed signal $\boldsymbol{y_c(t)}$, which can be expressed as

$$y_c(t) = x_c(t - \tau) \tag{2.1}$$

The Fourier transform $\boldsymbol{X_c(\omega)}$ of the continuous-time input signal is expressed as

$$X_c(\omega) = \int_{-\infty}^{\infty} x_c(t) e^{-j\omega t}\, \mathrm{d}t \tag{2.2}$$

where ω is defined to be the angular frequency of $2\pi f$ in radians.

The Fourier transform $\boldsymbol{Y_c(\omega)}$ of the delayed output signal is then expressed as

$$\begin{aligned} Y_c(\omega) &= \int_{-\infty}^{\infty} y_c(t) e^{-j\omega t}\,\mathrm{d}t \\ &= \int_{-\infty}^{\infty} x_c(t-\tau) e^{-j\omega t}\,\mathrm{d}t \\ &= e^{-j\omega\tau} X_c(\omega) \end{aligned} \tag{2.3}$$

The transfer function $\boldsymbol{H_c(\omega)}$ of the delay system can be expressed as

$$\begin{aligned} H_c(\omega) &= \frac{Y_c(\omega)}{X_c(\omega)} \\ &= \frac{e^{-j\omega\tau} X_c(\omega)}{X_c(\omega)} \\ &= e^{-j\omega\tau} \end{aligned} \tag{2.4}$$

As a result, $e^{-j\omega\tau}$ is the Fourier transform of the delaying the input by τ.

2.1.2 Discrete-time of Fractional Delay Systems

Consider the discrete-time bandlimited signal $\boldsymbol{x(n)}$ input into a delay system by a positive delay of D. The delayed output of the system is expressed as

$$y(n) = x(n-D) \tag{2.5}$$

In this context, D can only be considered as an integer value for the expression above to be valid. However, if the delay value D is an irrational number, the output value $\boldsymbol{y(n)}$ would have to be between two sampled values. Thus, the values of $\boldsymbol{y(n)}$

would have to be interpolated from the values of $\boldsymbol{x(n)}$.

Fractional delay of a signal refers to the typically irrational valued delay between two sampling points of an input signal. For these fractional delay filter applications, the fundamental issue is interpolating an input signal's values arbitrarily between two sampling points using varying mathematical methods. The delay value D can be represented as a sum of its integer part and fractional part:

$$D = \lfloor D \rfloor + d \tag{2.6}$$

where d refers to the fractional part of the delay, and $\lfloor D \rfloor$ is the greatest natural number less than D.

Analyzing the discrete-time input signal $\boldsymbol{x(n)}$, the discrete-time Fourier transform (DTFT) of the input signal is defined as [16]

$$X(\omega) = \sum_{n=-\infty}^{\infty} x(n) e^{-j\omega n} \, |\omega| \leq \pi \tag{2.7}$$

where $w = 2\pi fT$ is the normalized angular frequency, and T is the sampling interval.

The DTFT of the output signal can then be expressed as

$$Y(\omega) = \sum_{n=-\infty}^{\infty} y(n) e^{-j\omega n} = \sum_{n=-\infty}^{\infty} x(n-D) e^{-j\omega n} = e^{-j\omega D} X(\omega) \tag{2.8}$$

yielding a transfer function of

$$H_D(\omega) = \frac{Y(\omega)}{X(\omega)} = \frac{e^{-j\omega D} X(\omega)}{X(\omega)} = e^{-j\omega D}, |w| \leq \pi \tag{2.9}$$

As expected, the continuous-time and discrete-time transfer functions are equivalent with exception to the circular angular frequency.

Transforming Equation (2.9) to the Z-domain, the transfer function is transformed to

$$H(z) = \frac{Y(z)}{X(z)} = \frac{z^{-D}X(z)}{X(z)} = z^{-D} \tag{2.10}$$

where D is a real-valued number representing the length of the delay in samples and

$$z = e^{j\omega} \tag{2.11}$$

for the Z-transform.

Analyzing the magnitude and phase response of this transfer function, the magnitude is unity for all frequencies with a linear phase response with a slope $-D$. This system is generally referred to as an allpass system with linear phase response.

$$|H(z)| = \left|H(e^{j\omega})\right| = 1 \tag{2.12}$$

and

$$arg\{H(e^{j\omega})\} = -D\omega \tag{2.13}$$

Based upon these characteristics, it is evident that the ideal delay system includes all of the frequency components of the input signal with the same delay D.

To design a filter that can fractionally delay an input signal in a discrete-time application, some form of interpolation must be used to produce correct the output

values $\boldsymbol{y(n)}$ that are in between the samples of $\boldsymbol{x(n)}$. For the following techniques to work, the input signal must be bandlimited to half the sampling rate.

2.1.3 Signal Reconstruction

In an ideal interpolating application, to compute a fractionally delayed discrete-time signal, the amplitude of the actual continuous-time signal $\boldsymbol{x(t)}$ must be computed for all t. Shannon's signal reconstruction formula [17] from the sampling theorem can be used to reconstruct the continuous-time signal from obtained samples. Assuming an input signal that is bandlimited to half of the sampling frequency, the reconstruction formula is valid and expressed as

$$x(t) = \sum_{n=-\infty}^{\infty} x(nT) \frac{\sin[\frac{\omega_s}{2}(t-nT)]}{\frac{\omega_s}{2}(t-nT)} = \sum_{n=-\infty}^{\infty} x(nT) sinc[\frac{\omega_s}{2}(t-nT)] \tag{2.14}$$

where $\boldsymbol{x(t)}$ is a continuous-time signal, ω_s is 2π multiplied by the sampling frequency, and T is the sampling interval. The normalized *sinc* function is defined as

$$sinc(t) = \frac{sinc(\pi t)}{\pi t} \tag{2.15}$$

Considering the reconstruction formula in Equation (2.9), the ideal bandlimited sinc interpolator has a continuous-time impulse response of:

$$h_c(t) = sinc(\frac{\omega_s t}{2\pi}) \tag{2.16}$$

This formula allows for the conversion of the discrete-time sampled input signal

into a reconstructed continuous-time signal. In order for this conversion formula to be relevant for the fractional delay application, the input sampled signal must be delayed by the desired delay D, where

$$D = D_{int} + d \tag{2.17}$$

D_{int} refers to an integer delay in terms of the number of samples, and d refers to the fraction delay between 0 and 1.

In order to obtain the delayed interpolated discrete time output, Equations (2.16) and (2.17) are considered for a reconstructed discrete-time signal that is shifted and re-sampled by the necessary delay parameter D in (2.18) [4].

$$y(n) = x(n - D) = \sum_{k=-\infty}^{\infty} x(k) sinc(n - D - k) \tag{2.18}$$

Comparing with Equations (2.16) and (2.18), the impulse response of this system (2.18) is a shifted and sampled implementation of the infinitely long *sinc* function, which yields a noncausal system. As a result, the following sections describe fractional delay filters that are implemented as approximations to the impulse response of (2.16).

2.1.4 Approximating FIR Filters

The following sections illustrate techniques that approximate the ideal fractional delay system and are implemented via FIR-based filters. The transfer function for an FIR

filter is expressed by

$$H(z) = \sum_{n=0}^{N} h(n) z^{-n} \tag{2.19}$$

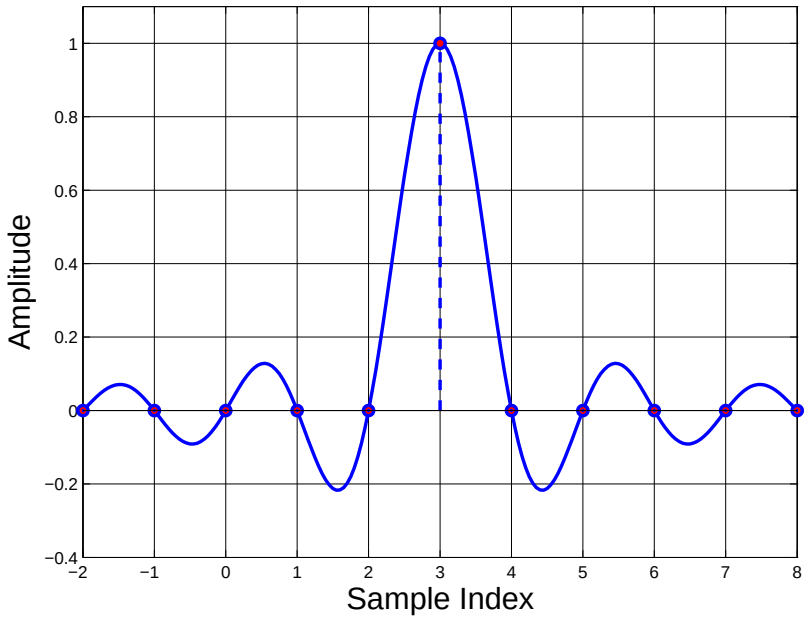
where N is the order of the filter, and $h(n)$ refers to the real-valued coefficients that form the impulse response of the filter. The main objective in each of these techniques is to minimize the error between the transfer function of the approximated filter technique and the transfer function of the ideal bandlimited sinc interpolator. This error function is expressed as

$$E(e^{j\omega}) = H(e^{j\omega}) - H_{ideal}(e^{j\omega}) \tag{2.20}$$

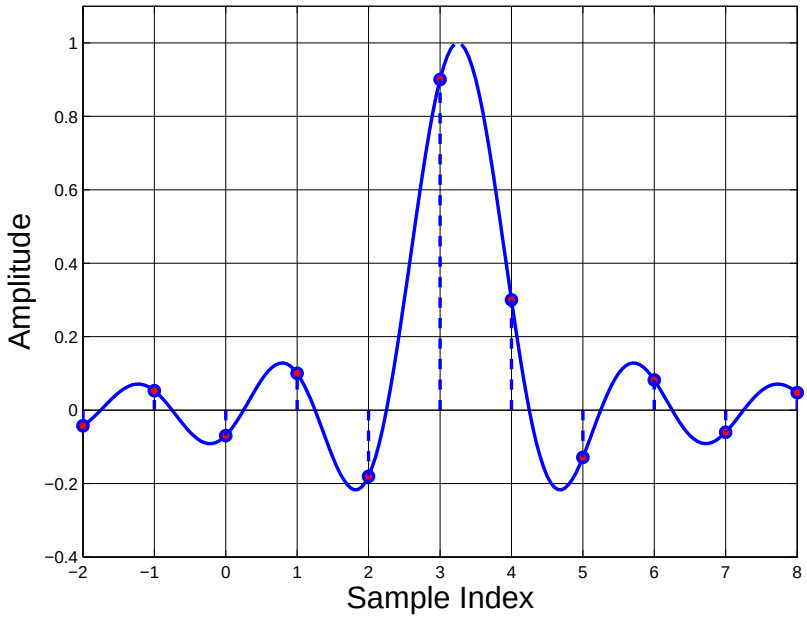
2.1.5 Accurately Implementing Causal Approximating Filters

Based upon Equation (2.14) in Section 2.1.3, it's evident that the ideal FD filter is an ideal bandlimited sinc interpolator. As a result, the main purpose of all of the FIR approximating FD filters is to minimize the approximation error between the FIR filter and the ideal bandlimited sinc interpolator.

The impulse response of an ideal bandlimited *sinc* interpolator is shown in Figure 2.1. Figure 2.1a depicts the ideal interpolator delayed by 3 samples. It's evident that when the *sinc* function is shifted by an integer delay, only the center value is non-zero. As expected, this interpolator, shifted by an integer delay, is equivalent to a cascade of unit delays. When the *sinc* interpolator is shifted by 3.25 samples, as shown in Figure 2.1b, each sampled value is non-zero, and the overall response is centered by the fractional delay value.



(a)



(b)

Figure 2.1: (a) *Sinc* Function Delayed by 3 Samples and (b) *Sinc* Function Delayed by 3.25 Samples.

Since FIR filters cannot center the impulse response exactly over the fractional delay value, FIR filters are designed with the fractional delay between the two central taps of an FIR filter for odd ordered filters or within half a sample from the central tap for even ordered filters to best match the response of a *sinc* interpolator. For this example, an FIR filter that approximates the response of this *sinc* interpolator requires a filter order of 6 or 7 to best center the fractional delay. After the filter has a fixed center point, the delay should follow the inequality:

$$\frac{N-1}{2} \leq D \leq \frac{N+1}{2} \tag{2.21}$$

where N is the order of the filter. The integer part of the delay D_{int} should follow the equation for odd ordered filters:

$$D_{int} = \frac{N-1}{2} \tag{2.22}$$

and

$$D_{int} = \begin{cases} \frac{N}{2}, & 0 \leq d < \frac{1}{2} \\ \frac{N}{2} - 1, & \frac{1}{2} \leq d \leq 1 \end{cases} \tag{2.23}$$

for even ordered filters [1]. Following Equations (2.21), (2.22), and (2.23), the filter will be designed with the fractional delay value within the central filter taps to best match the response of the *sinc* interpolator.

As fractional delay values vary away from the center point of the filter, fewer filter

coefficients correspond to the ideal impulse response of the *sinc* interpolator since the impulse response of the FIR will no longer be within the center. As a result, the overall response's accuracy would reduce as the fractional delay moves from the center of the FIR filter.

In addition to centering the filter over the fractional delay, the FIR filter's coefficients can only correspond to the *sinc* interpolator's values when $t \geq 0$ in order to remain causal. If the FIR filter coefficients are not truncated from the beginning of the ideal impulse response, the filter must follow certain restrictions to maintain causality. The index M of the first non-zero sample should correspond to

$$M = \begin{cases} round(D) - \frac{N}{2} & \text{for even ordered filters} \\ \lfloor D \rfloor - \frac{N-1}{2} & \text{for odd ordered filters} \end{cases} \tag{2.24}$$

where the FIR filter will only be causal if $M \geq 0$. If $M < 0$, an integer delay may need to be added to the system in order to create a causal filter. This characteristics will be further explored in the proposed VFD FIR filter design in Chapter 3.

2.2 Truncated Sinc Interpolator

The first discussed technique is based upon an approximation of the sinc bandlimited interpolator. This method seeks to minimize the least squared (LS) error function $\boldsymbol{ELS}$ which is equal to the integrated squared magnitude of the error function's frequency response in Equation (2.27) as

$$ELS = \frac{1}{\pi} \int_0^{\pi} \left| H(e^{j\omega}) - H_{ideal}(e^{j\omega}) \right|^2 \, \mathrm{d}\omega \tag{2.25}$$

The LS of the error function can be converted back into the sampled time domain via Parseval's theorem,

$$\sum_{n=-\infty}^{\infty} |a_n|^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} |A(x)|^2 \, \mathrm{d}x \tag{2.26}$$

to

$$ELS = \sum_{n=-\infty}^{\infty} |h(n) - h_{ideal}(n)|^2 = \sum_{n=-\infty}^{\infty} [h_{ideal}(n)^2 + h(n)^2 - 2h(n)h_{ideal}(n)] \tag{2.27}$$

The first term of the sum,

$$\sum_{n=-\infty}^{\infty} h_{ideal}(n)^2 \tag{2.28}$$

can again be simplified using Parseval's theorem and the ideal transfer function of Equation (2.19) in the following way

$$\sum_{n=-\infty}^{\infty} h_{ideal}(n)^2 = \frac{1}{\pi} \int_0^{\pi} \left| H_{ideal}(e^{j\omega}) \right|^2 \, \mathrm{d}\omega = \frac{1}{\pi} \int_0^{\pi} \left| e^{j\omega}) \right|^2 \, \mathrm{d}\omega = 1 \tag{2.29}$$

Using Equation (2.28) and knowing that $h(n)$ refers to coefficients when $n \geq 0$, the closed form solution of $\boldsymbol{ELS}$ is

$$ELS = 1 + \sum_{n=0}^{N} [h^2(n) - 2h(n)sinc(n - D)] \tag{2.30}$$

Regarding the closed form solution above, it is evident that the optimal $N + 1$ coefficients of an N-th order FIR filter using this technique would be truncated

symmetrically around the central point of $h_{ideal}(n)$, and the impulse response $h(n)$ of the approximating filter is

$$h(n) = \begin{cases} sinc(n-D) & 0 \leq n \leq N \\ 0 & \text{otherwise} \end{cases} \tag{2.31}$$

The truncated and shifted *sinc* function is a fairly intuitive design for implementation via an FIR filter structure. However, the truncation of the ideal interpolator's impulse response causes a ripples in the frequency response of the approximation for both the magnitude and phase response. A more in depth study of the responses can be found in [15].

2.3 Asymmetric Windowed Sinc Interpolator

An alternative to the truncated sinc interpolator is to use a bell-shaped window function for weighting the coefficients in the time domain. This technique is implemented to better control the performance of specific frequency bands of the input signal. This design is derived in more detail in [18].

The impulse response of the asymmetric windowed sinc interpolator FIR filter is expressed as

$$h(n) = \begin{cases} w(n-D)sinc(n-D) & M \leq n \leq M+N \\ 0 & \text{otherwise} \end{cases} \tag{2.32}$$

where the $w(n)$ is the window function, and index M follows

$$M = \begin{cases} round(D) - \frac{N}{2} & \text{for even ordered filters} \\ \lfloor D \rfloor - \frac{N-1}{2} & \text{for odd ordered filters} \end{cases} \tag{2.33}$$

where N is the order of the filter, and D is the desired delay.

The midpoint of the *sinc* function and window function are shifted by D. As a result, the shifted *sinc* function will be windowed symmetrically with the respective window function. Several windowing functions designed in the time domain allow the fractional delayed shift by D, as shown in [7].

However, using this windowing function method will result in an increase in approximation error $\boldsymbol{ELS}$ since the windowing function does not seek to minimize the LS error. This design is targeted to have a lower ripple than the truncated sinc interpolator detailed in Section 2.2.

2.4 Lagrange Interpolator

The Lagrange interpolator is a technique that builds upon the traditional Lagrange polynomial interpolation model. This design is an FIR-based FD filter that has a constant magnitude response around the desired frequency band, i.e. a maximally-flat FD filter. It is an accurate approximation of the ideal bandlimited interpolator at low frequencies [1]. As described by Kootsookos et al., the maximally-flat design of an FD filter and approximating windowing function are equivalent to the Lagrange interpolation structure [10].

2.4.1 Polynomial Interpolation

Lagrange interpolation is based upon N-th order polynomial approximation using $N+1$ equally spaced samples of a function or signal. These $N+1$ samples can create a Lagrange approximation polynomial, or just Lagrange polynomial, which can be used to interpolate values between the sampled points. The form of a Lagrange polynomial for a fractional delay application follows as

$$y^*(D) = \sum_{n=0}^{N}[h(n,D) * x(n)] \tag{2.34}$$

where $n = 0, 1, 2, ..., N$, D is a fractional delay between 0 and N, and $h(n, D)$ refers to polynomials of parameter D as

$$h(n,D) = \prod_{k=0,k\neq n}^{N} \frac{D-k}{n-k} \quad \text{for } n = 0, 1, 2, ..., N \tag{2.35}$$

The true benefit to this formula is that these filter coefficients are real-valued and are computationally efficient to compute.

The following section will derive a general maximally-flat filter in the frequency domain and prove equivalence to the traditional Lagrange interpolation formula[15].

2.4.2 Derivation of Maximally-Flat Filter in Frequency Domain

To derive the transfer function of an N-th order maximally-flat filter, the error function

$$E(e^{j\omega}) = H(e^{j\omega}) - H_{ideal}(e^{j\omega}) \tag{2.36}$$

and its N derivatives are set to zero at an initial frequency ω_0, expressed as

$$\left.\frac{d^k E(e^{j\omega})}{d\omega^k}\right|_{\omega=\omega_0} = 0 \quad \text{for } k = 0, 1, 2, ..., N \tag{2.37}$$

This equation can then be converted back into

$$\left.\frac{d^k}{d\omega^k}\left[\sum_{n=0}^{N} h(n)e^{-j\omega n} - e^{-j\omega D}\right]\right|_{\omega=\omega_0} = 0 \quad \text{for } k = 0, 1, 2, ..., N \tag{2.38}$$

This expression is now differentiated following the index of k and uses $\omega_0 = 0$. Starting with index $k = 0$,

$$\sum_{n=0}^{N} h(n) - 1 = 0 \rightarrow \sum_{n=0}^{N} h(n) = 1 \tag{2.39}$$

Regarding this result, the equation shows that the sum of all of the FIR filter coefficients must equal unity. Thus, the magnitude response at $\omega = 0$ must be 1, consistent with a maximally-flat filter design.

Continuing with the index $k = 1$, the differentiation produces

$$-\sum_{n=0}^{N} jnh(n) + jD = 0 \rightarrow \sum_{n=0}^{N} nh(n) = D \tag{2.40}$$

and differentiating with index k = 2 produces

$$-\sum_{n=0}^{N} n^2 h(n) + D^2 = 0 \rightarrow \sum_{n=0}^{N} n^2 h(n) = D^2 \tag{2.41}$$

This differentiation continues until $k = N$. The $N+1$ linear equations that follow from Equation (2.37) result in a set of equations of the form below

$$-\sum_{n=0}^{N} n^k h(n) = D^k \quad \text{for } k = 0, 1, 2, ..., N \tag{2.42}$$

The set of $N + 1$ equations can be then expressed in the matrix form as

$$Vh = v \tag{2.43}$$

where V is an $(N + 1)$x$(N + 1)$ Vandermonde matrix, which is of the form

$$\mathbf{V} = \begin{pmatrix} 0^0 & 1^0 & 2^0 & \cdots & N^0 \\ 0^1 & 1^1 & 2^1 & \cdots & N^1 \\ 0^2 & 1^2 & 2^2 & \cdots & N^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0^N & 1^N & 2^N & \cdots & N^N \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 0 & 1 & 2 & \cdots & N \\ 0 & 1 & 4 & \cdots & N^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 1 & 2^N & \cdots & N^N \end{pmatrix} \tag{2.44}$$

h is a vector of the FIR filter coefficients,

$$\mathbf{h} = \begin{bmatrix} h(0) & h(1) & h(2) & \cdots & h(N) \end{bmatrix}^T \tag{2.45}$$

and v is a delay vector.

$$\mathbf{v} = \begin{bmatrix} 1 & D & D^2 & \cdots & D^N \end{bmatrix}^T \tag{2.46}$$

Since the Vandermonde matrix is nonsingular [19], solving for the coefficient vector expression yields

$$h = V^{-1}v \tag{2.47}$$

and a closed form representation of the FIR filter coefficients [20] is expressed as

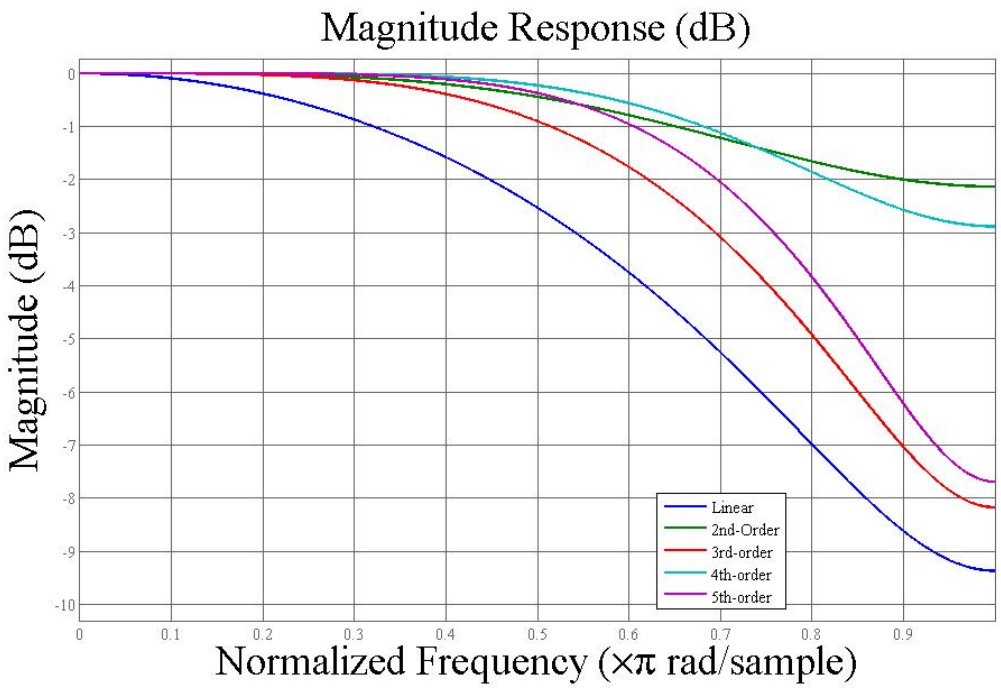
$$h(n) = \prod_{k=0,k\neq n}^{N} \frac{D-k}{n-k} \quad \text{for } n = 0, 1, 2, ..., N \tag{2.48}$$

Equation (2.48) is equivalent to the Lagrange interpolation formula for equally spaced samples presented in the time-domain [10].

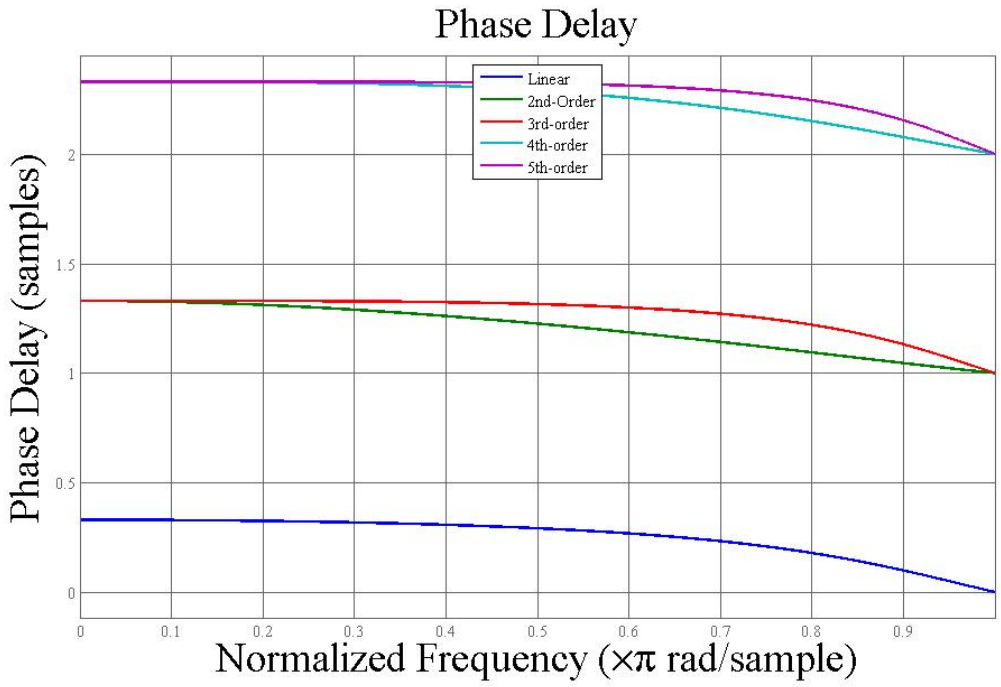
2.4.3 Filter Responses of Lagrange Interpolation

Regarding the practical performance of Lagrange interpolation, several orders of a Lagrange interpolation-based FIR filter was simulated for magnitude and phase response. The following plots illustrate the magnitude and phase response of a linear, second-order, third-order, fourth-order, and fifth-order filter.

A fixed delay parameter of 0.33 was used for each of the filter responses. Also, the normalized frequency 0.5 corresponds to the Nyquist frequency. As shown in the Figure 2.2a, the magnitude from a normalized frequency of 0 to 0.5 approaches the unity magnitude. However, comparing the odd ordered filters with the even ordered filters, it is evident that the even order filters do result in better performance than the odd order filters. Both filters do still produce the desired flat response. Regarding the phase delay response in Figure 2.2b, there are no ripples in the response as expected. The phase delay is consistent across each of the filter orders as well and exhibits the



(a)



(b)

Figure 2.2: (a) Magnitude and (b) Phase Delay of Langrange Filters

response of a linear-phase FIR filter.

2.4.4 Algorithmic Computation of Lagrange Interpolation

Following the equation for computing the Lagrange interpolator coefficients,

$$h(n) = \prod_{k=0,k\neq n}^{N} \frac{D-k}{n-k} \quad \text{for } n = 0, 1, 2, ..., N \tag{2.49}$$

it is evident that each of the coefficients are $N-th$ order polynomials parameterized by the delay D. Computing an example of an 2nd-order filter,

$$\begin{aligned} h(0) &= \tfrac{1}{2}(D-1)(D-2) \\ h(1) &= -(D)(D-2) \\ h(2) &= \tfrac{1}{2}(D)(D-1) \end{aligned} \tag{2.50}$$

As a result, computing these coefficients requires two addition and two multiply operations per coefficient. In a general $N-th$ order Lagrange interpolator, N addition and N multiply operations are required for each coefficient. In total, N^2+N addition operations and N^2+N multiply operations are required for the $N+1$ coefficients. However, observing the example above, it is clear that there are several repeated calculations performed, and this can be exploited in the design of the filter.

Further observing the structure of the coefficient computation, it is evident that many calculations must be performed to update each coefficient. This may be difficult in a given application and platform to update the coefficients if the delay parameter varies, as with most relevant applications. Typically, the coefficient would have to be updated within the sampling window of the input signal, which may or may not

be possible based upon the platform. The following structure improves upon this potential limitation, allowing variable fractional delay.

2.5 Farrow Structure

2.5.1 Overview

The Farrow structure is presented as an FIR-based structure that better allows for variable fractional delay. This structure was first designed by C. W. Farrow of AT&T labs in [?] for implementation on a WE DSP20 digital signal processor for echo cancellation on modems.

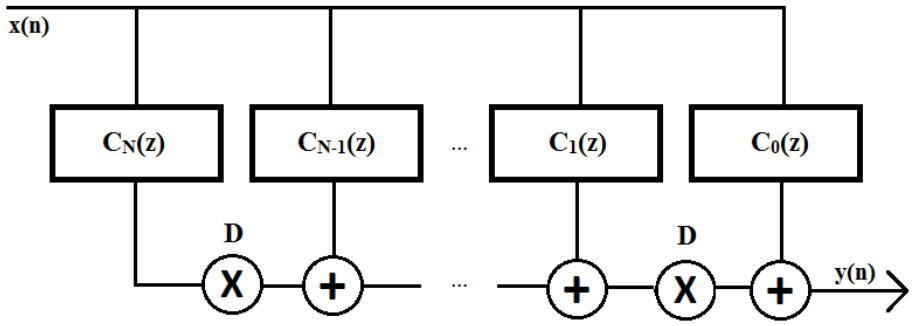


Figure 2.3: Farrow Structure of $N+1$ FIR Filters

The Farrow structure is represented as $N+1$ FIR filters with constant coefficients connected into one filter, as shown in Figure 2.3. To best handle a varying fractional delay D, the coefficients of the Farrow structure are not parameterized by the delay D. As a result, the coefficients of the FIR filters are fixed for a given filter structure order even if the fractional delay varies.

As shown in the figure above, the delay parameter is multiplied and accumulated to the outputs of the FIR filter banks. As a result, the total interpolation polynomial

resulting at the output of the whole filter can be updated much faster as the delay changes as the delay values are not pipelined deeply throughout the whole structure. This structure was based upon the design of each filter coefficient of the FIR interpolating filter be expressed as an $N-th$ order polynomial based on the delay parameter D. As a result, the Farrow structure can be implemented as a variable fractional delay filter of $N+1$ FIR filters with constant coefficients.

2.5.2 Derivation of Farrow Structure

To derive the closed form solution of the Farrow structure, the interpolation problem is addressed in the z-domain as follows

$$Y(z) = H(z)X(z) \tag{2.51}$$

where $\boldsymbol{Y(z)}$ and $\boldsymbol{X(z)}$ are the z-transforms of $\boldsymbol{y(n)}$ and $\boldsymbol{x(n)}$, respectively. The transfer function $\boldsymbol{H(z)}$ can be expressed as

$$H(z) = \sum_{k=0}^{N} C_k(z) D^k \tag{2.52}$$

Using the requirement for integer delays,

$$Y(z) = z^{-D}X(z) \rightarrow H(z) = z^{-D} = \frac{Y(z)}{X(z)} \quad \text{for } D = 0, 1, 2, ..., N \tag{2.53}$$

and Equation (2.52) leads to $N+1$ equations

$$\sum_{k=0}^{N} C_k(z) D^k = z^{-D} \quad \text{for } D = 0, 1, 2, ..., N \tag{2.54}$$

These equations may be expressed in matrix form as

$$UC = Y \tag{2.55}$$

where the $(N+1)$ by $(N+1)$ matrix U is the transpose of the Vandermonde matrix of equation [19],

$$\mathbf{U} = \begin{pmatrix} 0^0 & 0^1 & 0^2 & \cdots & 0^N \\ 1^0 & 1^1 & 1^2 & \cdots & 1^N \\ 2^0 & 2^1 & 2^2 & \cdots & 2^N \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ N^0 & N^1 & N^2 & \cdots & N^N \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 4 & \cdots & 2^N \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & N & N^2 & \cdots & N^N \end{pmatrix} \tag{2.56}$$

vector C is the coefficient vector,

$$\mathbf{C} = \begin{bmatrix} C_0(z) & C_1(z) & C_2(z) & \cdots & C_N(z) \end{bmatrix}^T \tag{2.57}$$

and Y is a delay vector.

$$\mathbf{Y} = \begin{bmatrix} 1 & z^{-1} & z^{-2} & \cdots & z^{-N} \end{bmatrix}^T \tag{2.58}$$

Thus, solving for the coefficient vector C yields

$$C = U^{-1}Y \tag{2.59}$$

where based upon [19], U is nonsingular and has an inverse matrix. For an easier reference, U^{-1} will be referred to as Q. Thus, Q can be expressed as

$$\mathbf{Q} = \begin{bmatrix} q_0 & q_1 & q_2 & \cdots & q_N \end{bmatrix}^T \tag{2.60}$$

As a result, the transfer functions $C_n(z)$ can be expressed as

$$C_n(z) = \sum_{k=0}^{N} q_n(k) z^{-k} \tag{2.61}$$

Thus, the coefficients $q_n(k)$ for the $N+1$ FIR filters $C_n(z)$ are then computed using the inverse of the transposed Vandermonde matrix. These coefficient derivation is often described as the Farrow structure of Lagrange interpolation.

2.5.3 Algorithmic Computation of Farrow Structure

The computation involved in constructing the Farrow structure is now described by a 3rd order filter structure. Building a 3rd order Farrow structure requires the use of

Equation (2.55) with $N = 3$. This equation is then expressed as

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 3 & 9 & 27 \end{bmatrix} \begin{bmatrix} C_0(z) \\ C_1(z) \\ C_2(z) \\ C_3(z) \end{bmatrix} = \begin{bmatrix} 1 \\ z^{-1} \\ z^{-2} \\ z^{-3} \end{bmatrix} \tag{2.62}$$

Inverting matrix U yields

$$Q = U^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -\frac{11}{6} & 3 & -\frac{3}{2} & \frac{1}{3} \\ 1 & -\frac{5}{2} & 2 & -\frac{1}{2} \\ -\frac{1}{6} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{6} \end{bmatrix} \tag{2.63}$$

with an overall transfer function $\boldsymbol{H(z)}$ of

$$H(z) = C_0(z) + C_1(z)D + C_2(z)D^2 + C_3(z)D^3 \tag{2.64}$$

As a result, computing N coefficients requires N parts of each of the N-th order FIR filters, resulting in $N(N+1)$ multiplications and N^2 additions for the sampled data calculations. Including the N multiplications and additions of the delay D, there are a total of N^2+2N multiplications and N^2+N additions per sample passing through the Farrow structure. However, observing the example above, it is clear that there are repeated calculations performed, and this can be optimized in the design of the structure.

Observing the structure of the coefficient computation, it is evident that this structure has fixed coefficients that are only parameterized by the order of the filter. As a result, this filter has better variable fractional delay performance. With that in mind, this improved performance comes at the expense of increased algorithmic complexity and area. This technique would require a much larger area compared to an implementation of the Lagrange interpolator due to the multiple FIR filters needed. As a result, the proposed design in Chapter 3 details a design that builds on the Lagrange interpolator rather than a Farrow structure.

Chapter 3: Proposed Design of VFD Filter on FPGA

This chapter will propose a VFD filter targeted for reconfigurable devices, such as a Field Programmable Gate Array (FPGA), based on the Lagrange interpolator. First, an overview of the proposed implementation of the VFD filter on reconfigurable hardware is presented. Following this discussion, the main design requirements are then detailed. Using these filter design requirements, design methods and differentiating features of the FPGA-based VFD filter is described, namely order-scalability. Two proposed paradigms of the VFD filter targeted for a hardware platform are presented in this chapter, a baseline software/hardware hybrid filter and a pure hardware solution designed for an FPGA.

3.1 Overview of Proposed Filter

Understanding the limitations of Lagrange interpolator and Farrow structure of Chapter 2, a design of a computationally efficient interpolator that can adequately handle VFD is desired. Based upon [21], it is evident that the equivalent FIR coefficients are parameterized by the fractional delay as well as the order of the filter. Typically, in a DSP processor implementation of this interpolator, the re-computation of the coefficients due to changing the desired fractional delay is not guaranteed to meet the time window constraint between sampling the input signal. However, using FPGA hardware, the coefficients can be easily recomputed within the sampling window of the input signal. Moreover, for a cost-effective implementation, reconfigurable devices,

like FPGAs, can provide sufficient hardware to meet the required time constraint.

This proposed hardware filter also allows the flexibility to change the order of the filter in order to adhere to the guidelines of fractional delay set in Section 2.1.4. Following Equations (2.21) (2.22), and (2.23), the filter's order and coefficients can be changed in order to correspond to centering the filter over the fractional delay without incurring additional delay penalty using a higher-order filter. Furthermore, as detailed by Section 2.1.4, centering the filter minimizes the approximation error. For a given desired fractional delay and corresponding order of the filter, the proposed filter will scale in the filter order size and re-compute the coefficients accordingly. This order-scalable filter supports applications that allow sacrificing precision to meet a real-time requirement. As a result, a relatively large FIR Lagrange interpolator was designed that can be scaled down to support delays with a small integer delay, following Equations (2.22) and (2.23). It is important to note that using the targeted hardware platform, further scaling the order of the maximum size of the FIR VFD filter is a simple process.

The proposed VFD filter is based upon the Lagrange interpolator as a result of the reduced algorithmic complexity, as well as the comparatively complex structure of implementing a Farrow structure. As a Farrow structure scales in size, the form of the structure changes based upon recomputation of the FIR filters' coefficients. Moreover, as the order of the structure increases by N, the area requirements increase by N^2. As a result, the VFD filter is based upon Lagrange interpolation instead of the Farrow structure.

The proposed design is targeted for implementation on the Xilinx Spartan-6 and

Virtex-6 FPGAs using the Xilinx System Generator toolkit, as well as Xilinx ISE. The System Generator toolkit within the Simulink environment offers great flexibility in I/O with the FPGA hardware design. To make use of this flexible interface, a prototype of a software/hardware hybrid VFD filter was created in System Generator.

3.2 Design Requirements

Based upon the targeted platform of reconfigurable hardware, the VFD filter design was catered to a minimal delay or latency with relatively high accuracy at low frequencies. Using the advantages of reconfigurable hardware, this design also targeted the ease of scalability with low computational complexity. The requirements of this filter are also based upon the applications of typical VFD filters. The aim for this filter is to be maximally-flat with linear phase and have high accuracy at relatively low frequencies. These characteristics match best for a filter based upon the Lagrange interpolator.

As mentioned above, in order to best minimize the approximation error of the filter response, the filter must center the data correctly. As a result, there are two options to center the filter, add delay lines to the sampled data as necessary or change the order of the filter to center the valid data across the FIR filter. These choices sacrifice precision with delay time. Changing the order of the filter correctly to center the sampled data will correspond to real-time applications that desires low latency.

Using the Xilinx System Generator toolkit, as well as Xilinx ISE, to design the VFD filter, one may use fixed-point data for all of the calculations performed on hardware. A majority of the calculations performed are on fractional values. Thus,

the calculations may be performed on floating-point or fixed-point computational units. However, the fixed-point computation units were selected for the VFD filter to reduce the number of cycles of coefficient calculation as well as resource requirements. The Q-format representation of fixed-point signed fractional values are discussed in the following section.

3.2.1 Q-format Representation of Fixed-point Signed Fractional Values

The general expression for the signed fraction format, MQN bit format, can be expressed as:

$$D = -1^{b_N}2^M + b_{N-1}2^{M-1} + ... + b_{N-M-1}2^0 + b_{N-M-2}2^{-1} + ... + b_2 2^{-F+1} + b_1 2^{-F} \quad (3.1)$$

where, $F = N - 1 - M$, and the binary point is located after the 2^0 term. An N-bit two's complement binary numerical system can be ranged from $-2N-1$ to $2N-1-1$.

Equation (3.2) expresses the conversion of decimal number to MQN format binary number.

$$D2^F \rightarrow signedtwo'scomplementformat \quad (3.2)$$

Since fixed-point values are discrete, not all values can be expressed accurately with the MQN format. Smaller values will incur higher error in terms of percentage error when comparing large values of the same fixed-point format.

3.3 Designing FPGA-based Filters on System Generator

The Simulink environment wrapping the Xilinx System Generator toolkit is catered to designing hardware for relatively fast functional verification. The total design with the software interfacing into the FPGA hardware can be first simulated for functional testing. The hardware portion of the overall system can then be generated to a .bit file that can be uploaded into the FPGA. Using this hardware-in-the-loop test, the whole system can be tested on FPGA hardware using a software test interface.

The computational operations handled in either the software (Simulink) layer or the hardware (FPGA) layer are represented in Simulink and Xilinx blocks, respectively. The Simulink environment also reduces the hassle of interfacing between these computational blocks by abstracting the connections to the interfaces as general wires within each layer. In the software layer, Simulink allows the user to choose the data types between the computational blocks, while in the hardware layer, the System Generator toolkit allows for the data types to be boolean, fixed-point, or floating-point. As shown in Figure 3.1, the simulation environment is also separated between the software and hardware layers via the yellow Xilinx gateway block, which converts the data-types within the Simulink environment into the chosen data types of the hardware layer.

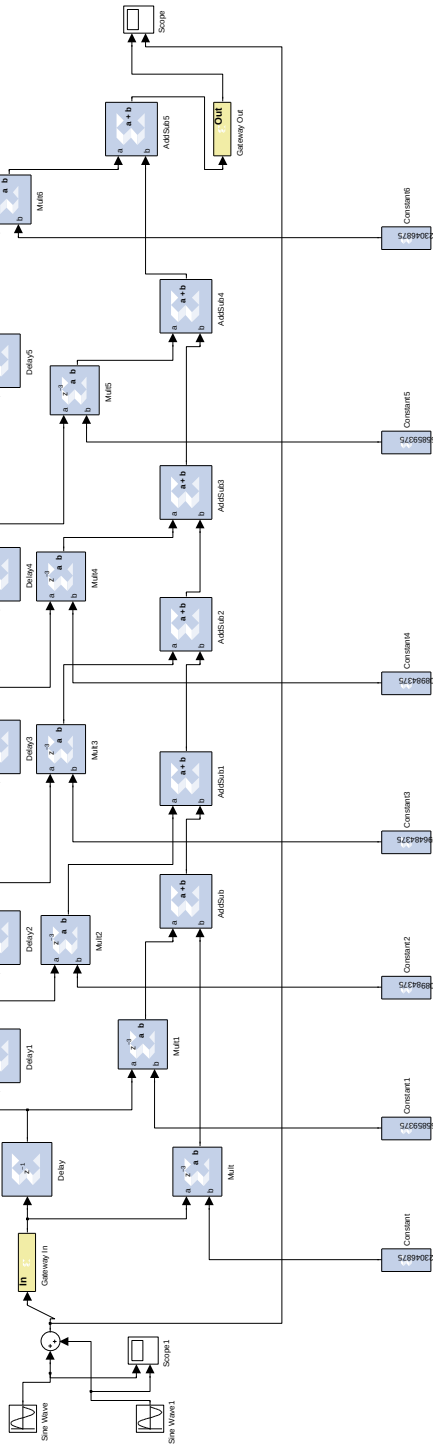


Figure 3.1: Sixth Order FIR Filter in the Simulink/System Generator Environment

The blue blocks within the Gateway In and Gateway Out blocks are of the Xilinx blockset for FPGAs, and the white blocks outside of the Xilinx gateway blocks are Simulink blocks. As shown in Figure 3.2, the two sine waves are Simulink sources that are being sent to the FPGA via the gateway in block, and the scope acts as a sink to view the data in the Simulink environment after processing by the hardware layer. These sources and sinks can be substituted for a variety of Simulink blocks, as well as sources and sinks from the Matlab workspace.

3.4 Software/Hardware Hybrid Filter

Designing the maximally-flat order-scalable VFD FIR filter in the System Generator toolkit, the baseline implementation was built as a software/hardware hybrid filter. The hybrid design is meant to make use of the flexibility of the accessible I/O interfaces within the Matlab Simulink environment and aims for functionality and testability over performance. This approach has the added bonus of being built for debugging purposes, as the whole structure is modularized between computing the coefficients and designing the FIR filter structure. The following figure outlines a high level view of the filter design structure.

3.4.1 Order-Scalable Structure

As shown in Figure 3.3, within the FIR filter structure of the hardware layer, each output tap of each of the adder computational units corresponds to the output of a FIR filter with a distinct order. Thus, each of these adder computational units are input into a signed fixed-point MUX block which has its output selected by the

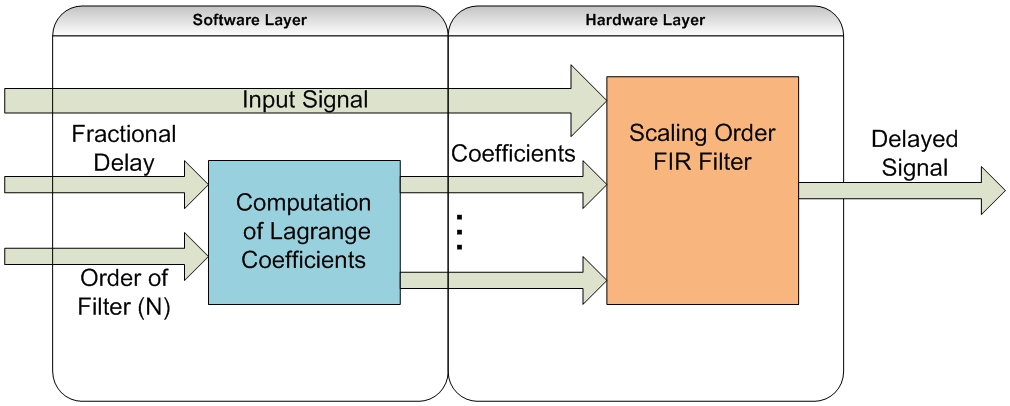


Figure 3.2: Overview of Software/Hardware Hybrid VFD Filter on FPGA

desired order of the filter.

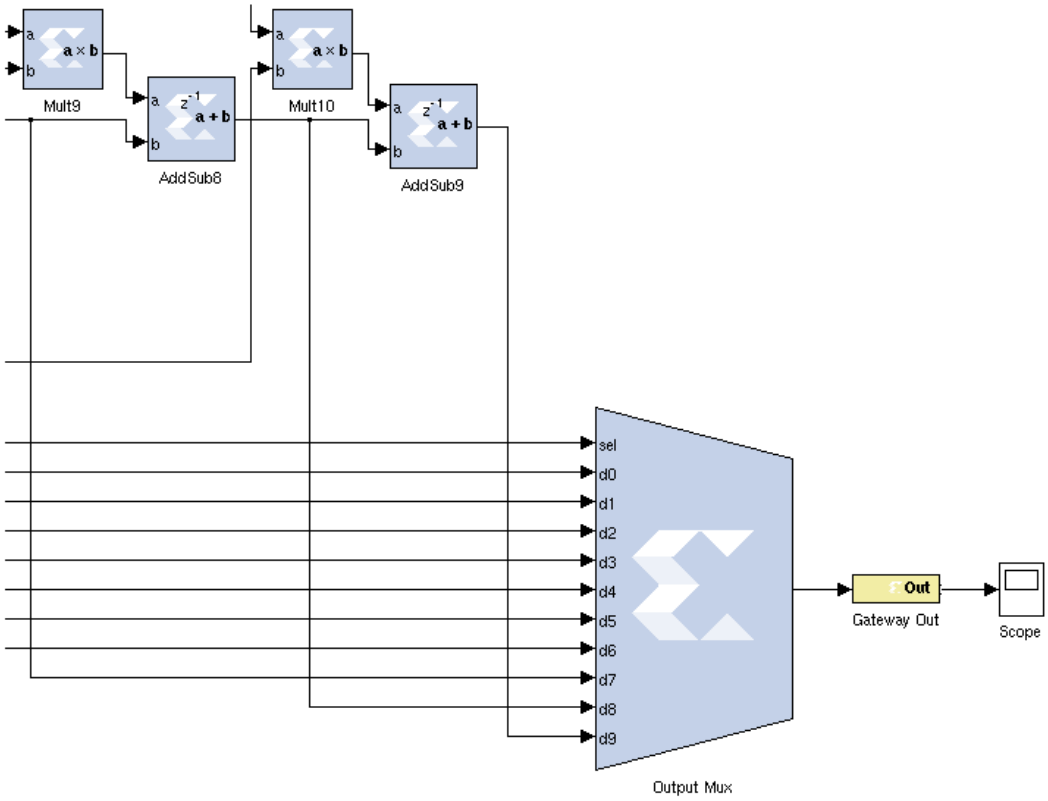


Figure 3.3: Multiplexing of Output Taps

By setting a fixed maximal size N for the overall filter structure, this design can output the desired responses for FIR filters of sizes 1 to N. Using the System Generator tookit, in this software/hardware hybrid filter, the order of the filter is presumed to be a value from software that is passed through a Xilinx gateway for proper QNM fixed-point form. Since the order of the filter, N, is an integer, the gateway will not format the value for fractional bits and will cause a fixed-point format of ceiling $\log_2(N+1)$ as an unsigned data type.

3.4.2 Software-based Lagrange Interpolator Coefficients Computational Unit

As shown on the software/hardware hybrid filter block diagram in Figure 3.2, the coefficients for the Lagrange interpolator are computed in the software layer. Using the System Generator toolkit, the Lagrange interpolator coefficients are computed using M-code, which is represented as a Simulink block in the Simulink environment wrapping the filter.

Using this environment, the coefficients are recalculated as the delay and order values change on the host CPU and are then passed through the Xilinx Gateways for fixed-point type conversion via the debug JTAG bus. These fixed-point values are then passed as the filter coefficients to the corresponds FIR filter coefficient slots as shown in Figure 3.4.

The pseudocode to compute these coefficients is presented below. This code follows the Lagrange coefficient equation derived in Section 2.3.2. This M-code was developed for a maximal FIR filter size of 10.

Algorithm 1 Compute Lagrange Coefficients

for $n = 0 \rightarrow N$ **do**
  $h \leftarrow 1$
  for $k = 0 \rightarrow N$ **do**
    if $k \neq n$ **then**
      $h \leftarrow h\frac{D-k}{n-k}$
    end if
  end for
  $H \leftarrow [H, h]$
end for
return H

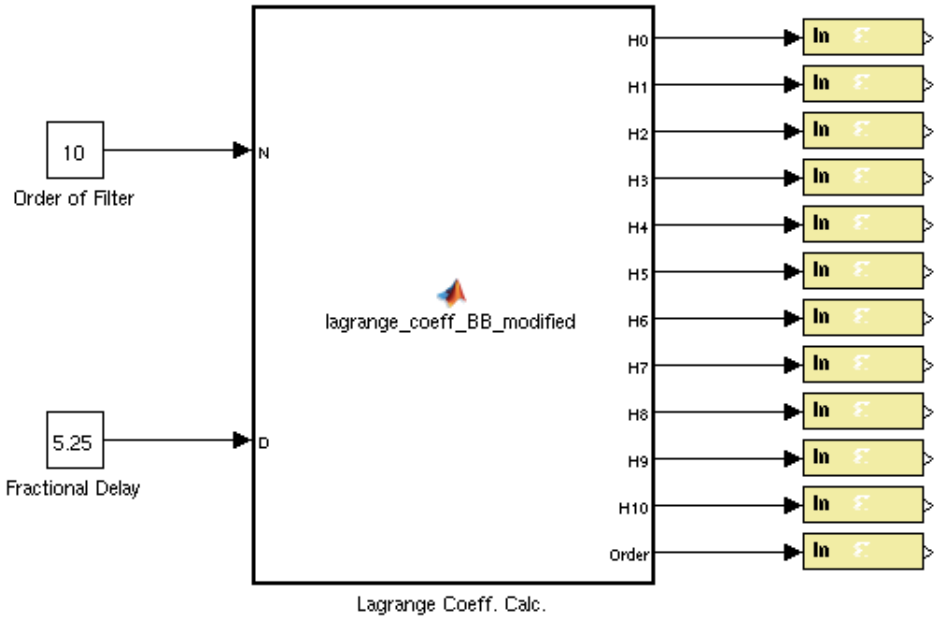


Figure 3.4: Matlab Coefficient Computational Block

3.5 Reconfigurable Hardware Filter

After the prototype of the software/hardware hybrid filter was designed and tested, a fully hardware filter was designed with the filter modularized between the computational unit of the Lagrange coefficients and the order-scalable FIR filter structure. As mentioned in the overview of the software/hardware hybrid filter, the FIR filter structure in the fully hardware filter is equivalent in both implementations. Figure 3.5 shows the coefficient computational unit within the hardware layer. The software layer provides the source and sink to functionally verify the correct operation of the overall filter. The computational unit of the Lagrange coefficients has been functionally verified. However, integrating this computational unit into the overall order-scalable Lagrange FIR filter remains the main focus of the future work of this project.

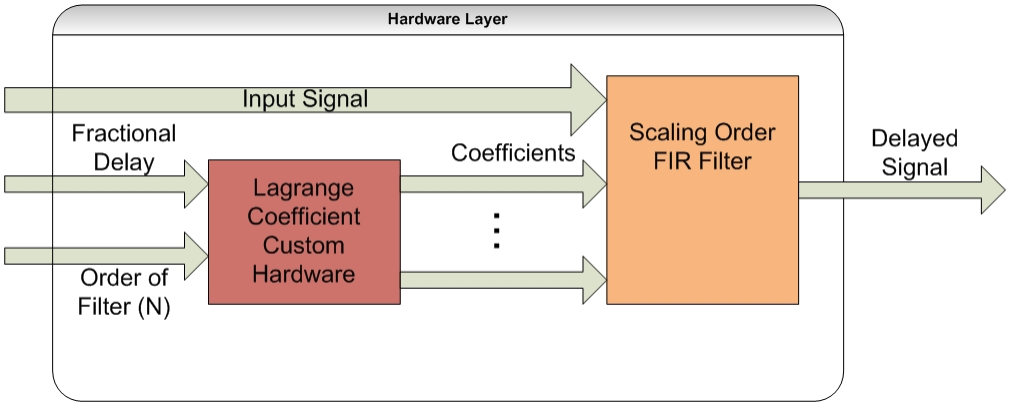


Figure 3.5: Fully Hardware-based Design of VFD FIR Filter on FPGA

3.5.1 Lagrange Interpolator Coefficients Hardware

The hardware VFD filter solution is integrated with either a parallel or serial custom Lagrange coefficient computational unit. Each computational unit was created using

VHDL in Xilinx ISE and functionally verified in ModelSim. The custom hardware computes the fixed-point filter coefficients based upon the order of the filter and the fractional delay as mentioned in the software/hardware hybrid design. The coefficient values are streamed into the respective coefficient registers of the maximum order FIR filter with a padding of zeros for each unused filter coefficient.

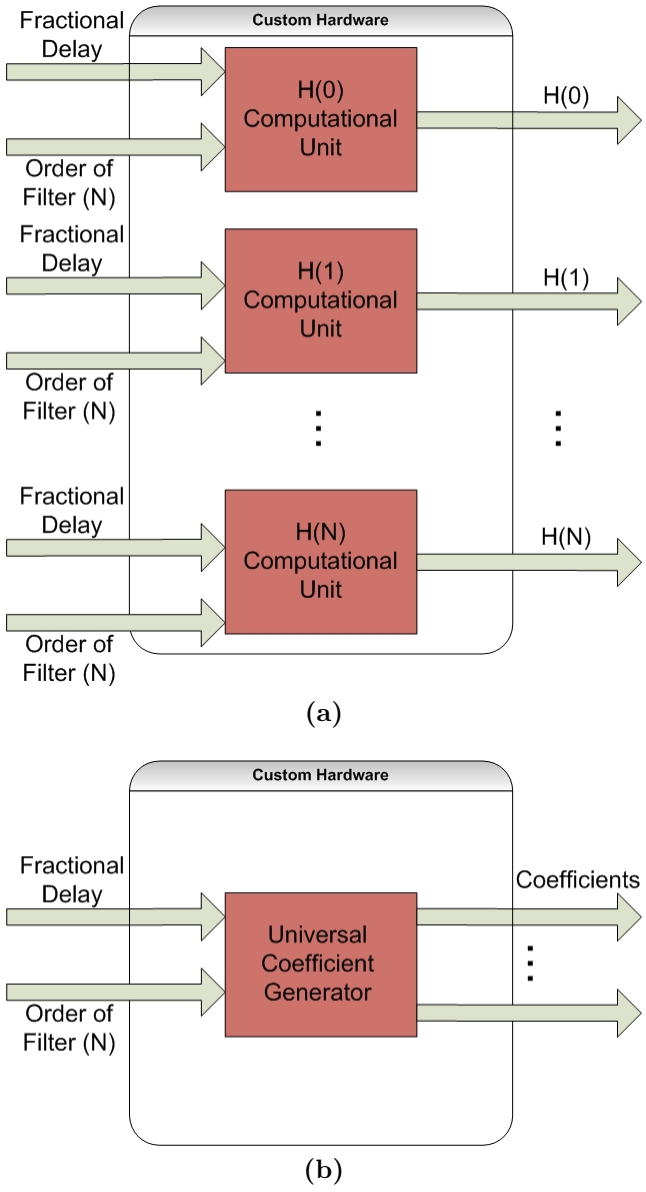


Figure 3.6: (a) Parallel Lagrange Coefficient Computation Unit and (b) Serial Lagrange Coefficient Computation Unit

Two hardware coefficient computational blocks were designed: a parallel unit and a serial unit. For the parallel unit, the overall computational unit for a maximal N-th order FIR filter includes $N+1$ coefficient computational units as described in Equation (2.35). As shown in Figure 3.6a, each individual coefficient computational unit has the filter order and the fractional delay as inputs, while the output of each coefficient computational unit is the respective coefficient. For the serial coefficient computation unit, as seen in Figure 3.6b, each of the $N+1$ coefficients of a N-th order FIR filter are computed serially with one signal computational unit with the same inputs and outputs of the parallel computational unit.

Each coefficient computational unit was coded in a behavioral design based upon the Lagrange Coefficient Equation (2.35) derived in Section 2.3.2. Each arithmetic operation on the data required MQN fixed-point operations. Each operation (add, subtract, multiply, and divide) in fixed-point yielded a different fractional bit precision. As a result, the internal signals across the arithmetic operations were of different fractional bit precisions. The fractional bit precision was optimized to maintain the highest bit precision through all of the necessary arithmetic operations.

Chapter 4: Experimental Setup and Results

This chapter will provide experimental setup and results of testing the designed and prototyped VFD filters. First, the performance analysis of the proposed and prototyped VFD filters are presented. The baseline software/hardware hybrid filter prototype results will be detailed and compared to a targeted high performance implementation. Following this discussion, the performance results of the hardware coefficient computational unit are then presented and evaluated. Finally, the costs and benefits of each type of hardware coefficient computational units are discussed.

4.1 Baseline Design Implementation in System Generator and Simulink

As described in Chapter 3, the baseline design for the VFD filter on FPGA hardware is the software/hardware hybrid filter. A prototype was produced using the System Generator and Simulink environment and tested for functional verification.

4.1.1 Sampled Input Data

Using the accessible I/O interfaces in the Simulink environment, the input signal processed in the software/hardware hybrid VFD filter was created by a combination of periodic signals. The data is in double floating-point format and is converted into the MQN fixed-point signed fraction format within the Xilinx gateway. During the testing process, 'chirp' and combinations of counters and sine waves were tested as inputs.

4.1.2 Coefficient Verification

As shown in Chapter 3, the baseline VFD filter incorporates a software-based coefficient computational unit. In the proof of concept prototype, the coefficients are computed using a Matlab computational block in the Simulink environment. The prototype has a maximal order of 10. Thus, each of the coefficients of the filters of order 1 through 10 are tested using a M-code script based on the Lagrange interpolation algorithm and verified for correctness.

4.1.3 Functional Verification of Input/Output Data

Using the prototype designed for the software/hardware hybrid VFD filter, the maximum size 10 order-scalable FIR filter was generated for a hardware-in-the-loop test as shown in Figure 4.1.

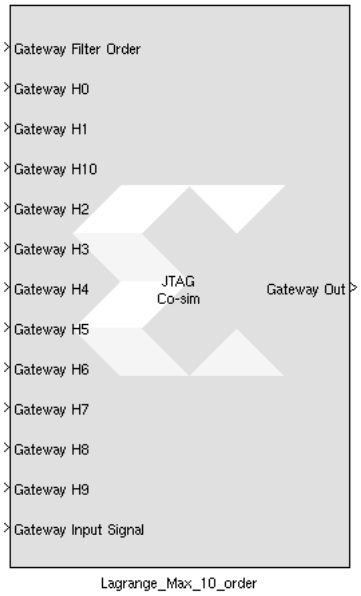


Figure 4.1: Generated Prototype of Scaling FIR Filter

The input data for verification as mentioned above is a combination of periodic waveforms. The values from the Matlab code block is computed in software on the host PC and passed through the JTAG interface to the fixed-point converting gateways within the generated hardware. The output of this order-scalable FIR filter is sent out of the gateways into a display scope and resampled to a Matlab vector.

Testing Fixed Fractional Delay

A first nominal test included keeping a fixed delay value throughout the hardware-in-the-loop test of 4.5 samples. Figure 4.2 displays the input signal and the interpolated delayed signal.

An ideal periodic wave of the tested input signal was generated in Matlab and delayed by 4.5 samples. This delayed ideal wave was compared to the delayed wave produced by the hardware-in-the-loop test.

Regarding Figure 4.3, it is evident that the actual delayed wave is approximately a sampled wave of the ideal delayed periodic wave. Figure 4.3b zooms into the ideal and measured waveforms and clearly illustrates that the measured and interpolated waveform tracks and samples the ideal waveform at each sample point. To estimate the performance of this actual delay wave, the percent error between the sampled points and sampled ideal wave was calculated to be approximately 1.88%.

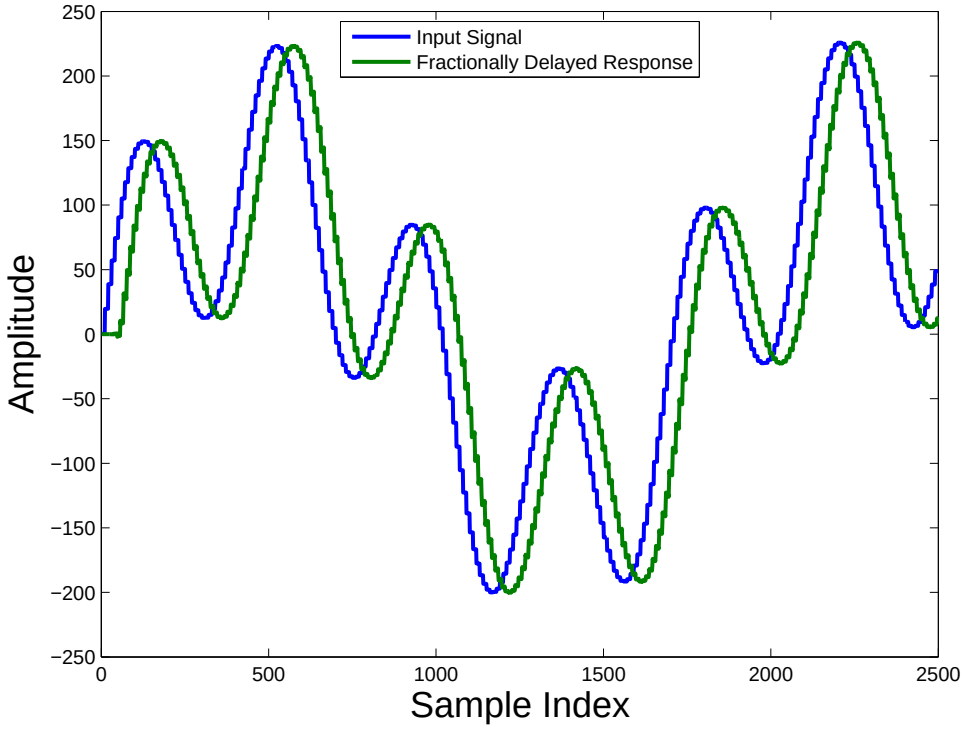
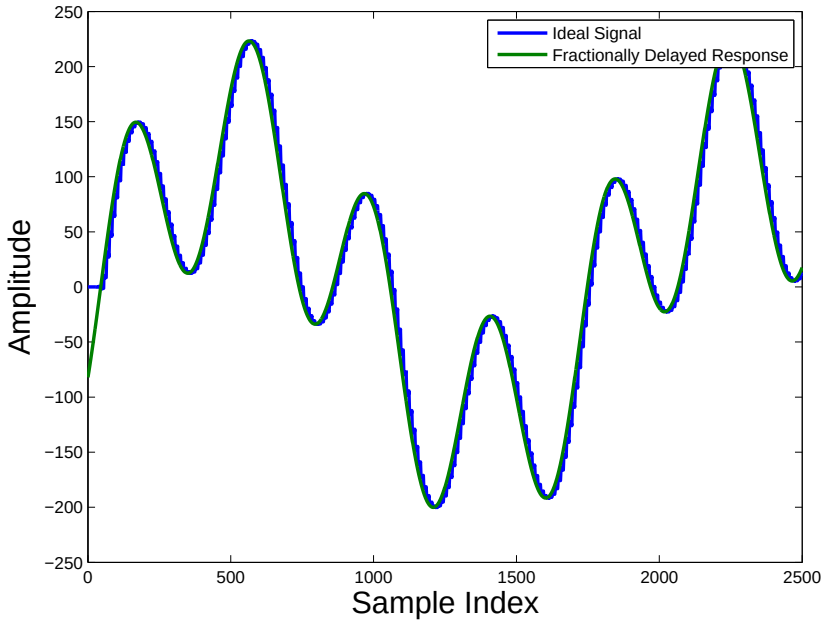
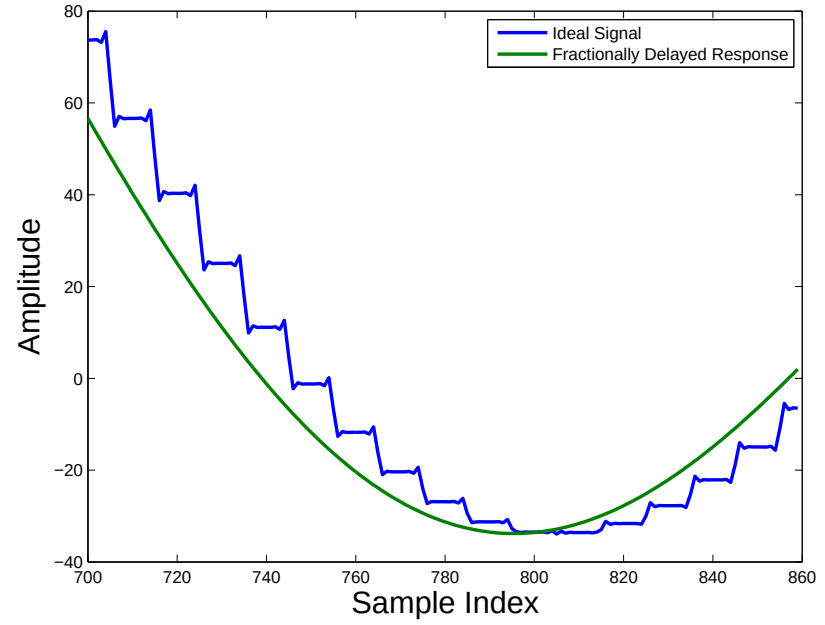


Figure 4.2: Test Run of Delaying Input Signal by 4.5 Samples



(a)

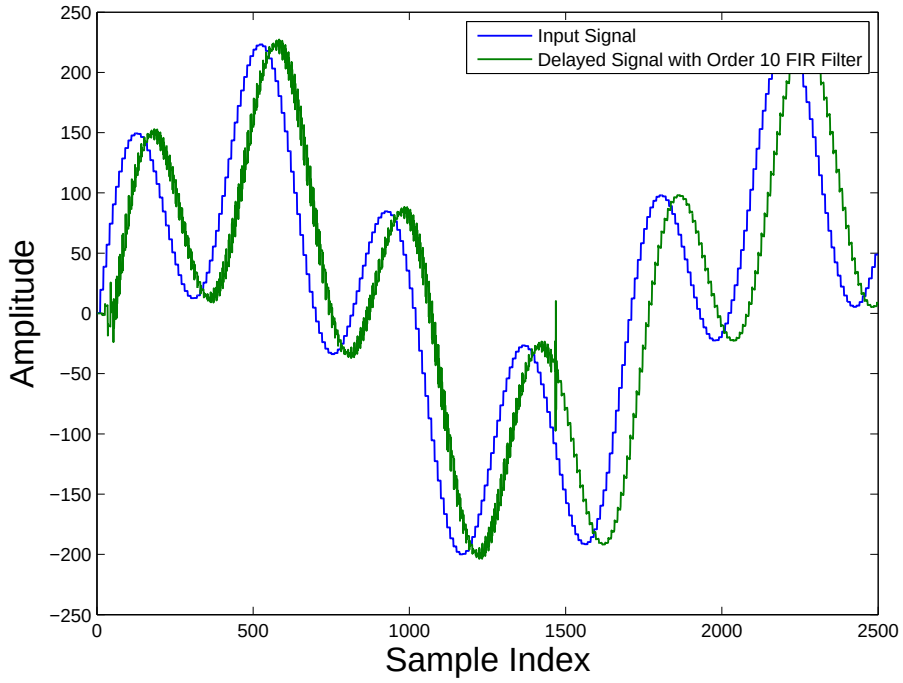


(b)

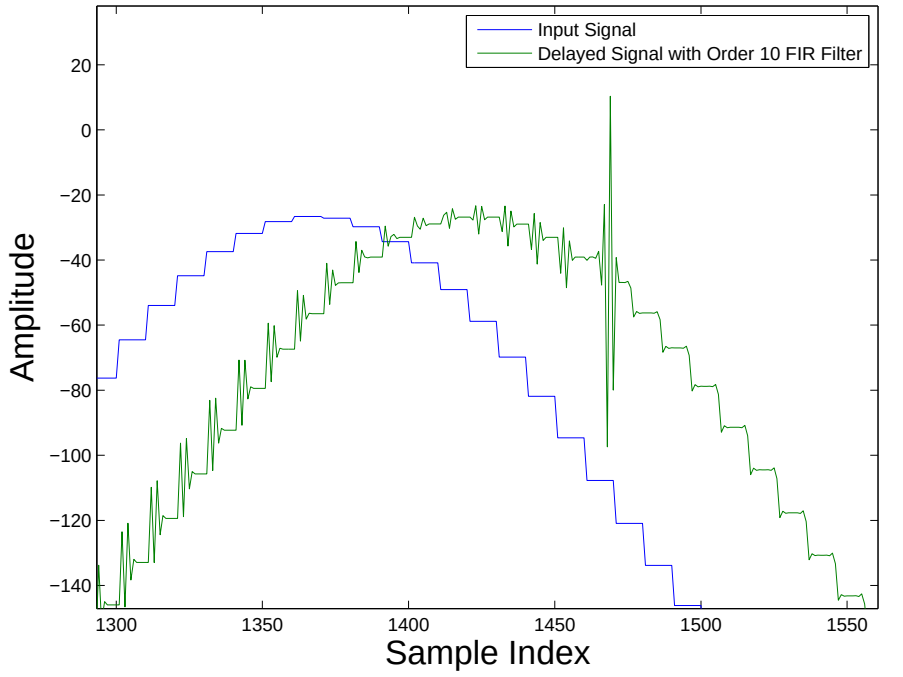
Figure 4.3: (a) Comparison of Ideal Delayed Input Signal and Actual Delayed Signal and (b) Zoomed in Comparison

The second test consisted of testing the order-scalability of the filter with fixed fractional delay. This test compared the responses of using a filter order that follows the fractional delay restrictions of Equations (2.21), (2.22), and (2.23) and the response of using a filter order that was outside of the restrictions.

Using the same input, with a fractional delay of 5.25 samples, the order of the filter was first set to 5. After pausing the simulation, the order of the filter was then set to 10, which corresponds to the order that centers the filter over the fractional delay of 5.25. As shown in Figure 4.4, the response of the 5th order filter was very noisy overall, and as shown in Figure 4.4b, half of each delayed sample was simply just fluctuations around the sampled value. After the order was changed to 10 at approximately the 1475 sample point, the filter reponse corresponds to the expected result of the fractional delay of the input signal. This test highlights the flexibility of this filter and is consistent with the discussion of minimizing the approximation error of Section 2.1.5.



(a)



(b)

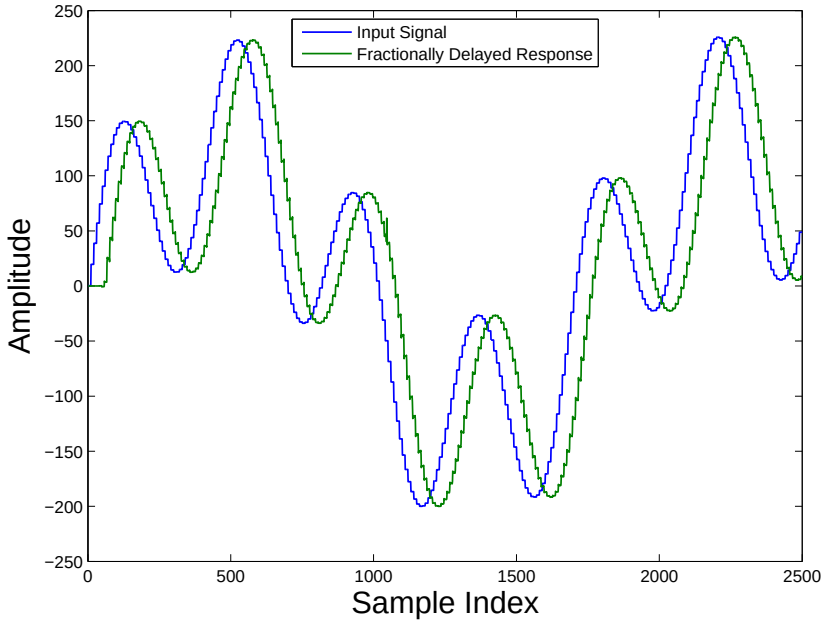
Figure 4.4: Test Run of Delaying Input Signal by 5.25 Samples

Testing Variable Fractional Delay

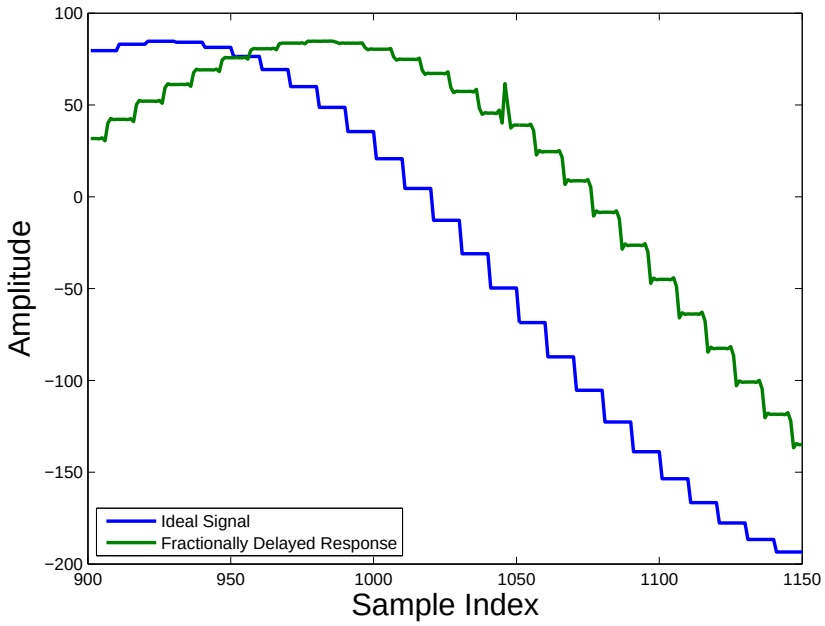
Two tests for variable fractional delay of the filter were performed: changing the fractional delay of the filter within the same FIR filter order, under the delay limitations presented in Section 2.1.5 and changing the fractional delay while respectively changing the order of the filter. In each case, one sample of the delayed signal is dropped, during which the coefficients must be recomputed and updated back into the FIR filter. During each of the trial runs, the simulation is paused while the order of the filter and the fractional delay values are changed. The simulation is then resumed with the new inputs values.

During the first test, a combination of periodic waves are supplied as the input, and the fractionally delayed wave was output with a FIR filter order of 10. The fractional delay during the beginning if the simulation was 4.75 samples, and after a pause in the simulation, the fractional delay was changed to 5.25 samples. As shown in Figure 4.5, at approximately 1050 samples into the simulation, one sample of the delayed wave was dropped during the recalculation of the Lagrange coefficients in software. However, following this dropped sample, the delayed wave again tracked the input wave with a 5.25 sample delay.

During the second variable fractional delay test, a combination of periodic waves are again supplied as the input, and the fractionally delayed wave was output. The simulation started with a filter order of 5 and a fractional delay of 2.35 samples. After a pause in the simulation, the filter order was changed to 10 with a fractional delay of 5.37 samples.

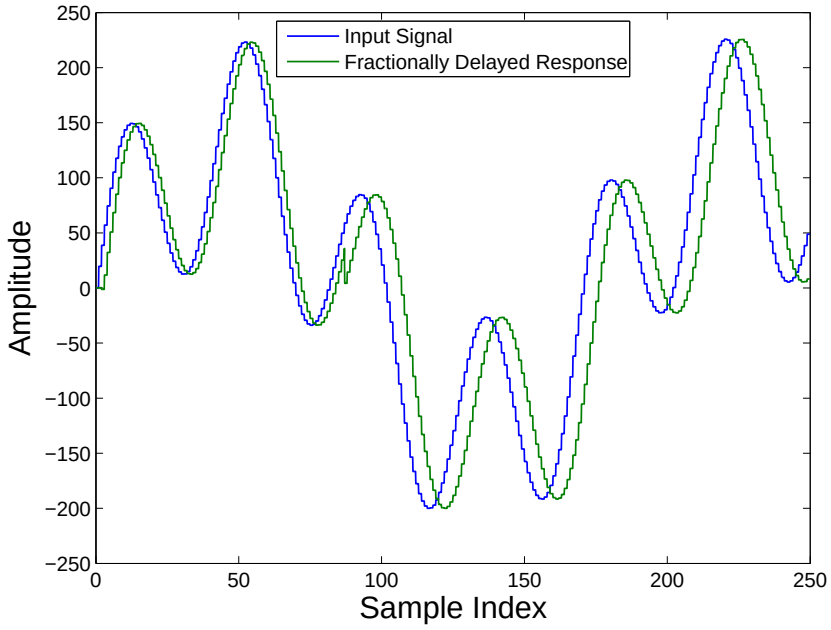


(a)

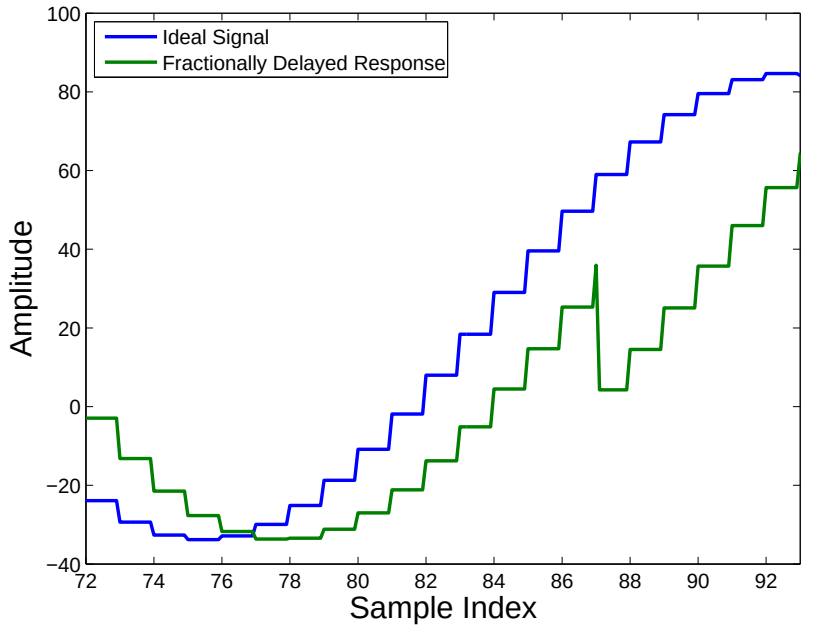


(b)

Figure 4.5: Test Run of Delaying Input Signal by 4.75 Samples then 5.25 Samples



(a)



(b)

Figure 4.6: Test Run of Delaying Input Signal by 2.35 Samples then 5.37 Samples

As shown in Figure 4.6, during the pause, there is a transition transient at approximately 87 samples into the simulation during the recalculation of the Lagrange coefficients. However, like the last trial run, the delayed wave starts to track the input signal right after the coefficients are recalculated with an updated fractional delay value.

4.1.4 Comparison to High Performance Implementation

The software/hardware hybrid VFD filter prototyped in the Simulink/System Generator environment was created for a proof of concept of the order-scalable VFD filter targeted for a hardware-based platform. However, with these designs in mind, the ideal platform for this software/hardware hybrid design is a general purpose processor connected to an order-scalable FIR filter within an FPGA via a high speed interconnect. One targeted platform for this design is the Xilinx ML507 system with a PowerPC 440 processor connected to a Virtex 5 FPGA via the PLB interface. A system with a hybrid hard processor core connected to reconfigurable hardware is the best platform for an interface that can cater to real-world VFD applications.

The PowerPC440 core would perform the processing of the Lagrange coefficients, and roughly estimating the performance of the Lagrange coefficient algorithm on the PowerPC RISC processor, computing 11 coefficients for the max 10 order FIR filter takes approximately 500 cycles assuming all fractional delay and filter order values are in registers. The computations will be performed serially for each coefficient and as a result, reducing the filter order will reduce the total computational time for the FIR filter. The PowerPC 440 can operate with a 400 MHz reference clock, while

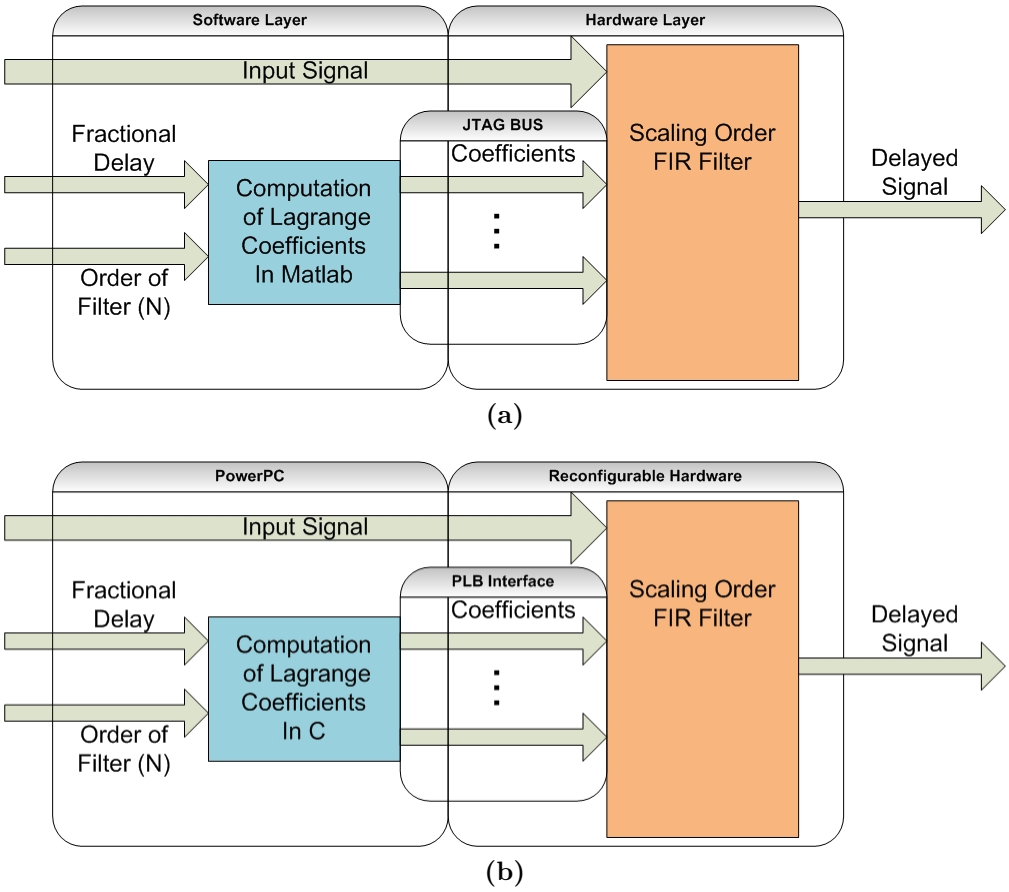


Figure 4.7: Overview of (a) Prototype and (b) High Performance Implementation

after each individual coefficient is computed, each coefficient can be transmitted to the FPGA via a 133 MHz processor local bus (PLB) interface.

Regarding the order-scalable FIR filter, Table 4.1 lists the approximated resource requirements of synthesizing the FIR filter on the Virtex 5 FPGA. LookUp Tables (LUTs) are the basic logic elements of the FPGA. The Input/Output Blocks (IOBs) are a grouping of basic elements that implement the input and output functions of an FPGA. The DSP48E1 is an embedded multiply-accumulate unit. In comparison to the available resources, this order-scalable FIR filter can easily be implemented

of the Virtex-5 FPGA. Furthermore, given the available resources, this filter can be scaled even higher than the initial maximal FIR filter order of 10.

Table 4.1: Approximated Resource Requirements of FIR Filter

Resource	*Number of Units*
Occupied Slices	479
Slice Registers/FF	160
Slice LUT	795
IOB	238
DSP48E1	11
% of Slices Used	4.27%

Overall, porting the design from the Simulink/System Generator environment will be part of the future work for this project. The Xilinx ML507 system or another processor core with reconfigurable hardware system are platforms that be used for real-world VFD applications using the Software/Hardware hybrid VFD filter detailed in this thesis.

4.2 Hardware Coefficient Computation Unit

Building upon the software/hardware hybrid filter, a parallel and serial Lagrange coefficient computational units were designed for an FPGA platform to be synthesized along with the order-scalable FIR filter as described in Section 3.5.1.

4.2.1 Coefficient Verification

The inputs to the coefficient computational units are the desired order of the filter and the fractional delay, which are both in the MQN fixed-point signed fraction format. The output of the computational units are the desired Lagrange coefficients in MQN fixed-point. To test the computational units, each desired filter order from 1 to 10

was tested in ModelSim with fractional delay values up to a precision of 10 fractional bits. The coefficients output from the computational unit were compared to computed values of a Matlab script that computes the algorithm of the Lagrange coefficients and verified with precision of up to 10 fractional bits.

4.2.2 Performance Analysis of Hardware Coefficient Computational Units

As detailed in Section 3.5.1, two hardware coefficient computation units were designed: a parallel computational unit and a serial computational unit.

Parallel Computational Unit

A parallel computational unit was synthesized to compute 11 coefficients for a maximum 10th order FIR filter. This block was synthesized and targeted for the Xilinx Virtex-6 FPGA. As 11 coefficients are computed in parallel, the parallel unit requires a relatively large amount of resources compared to the serial unit. Table 4.2 lists the approximated resource requirements of synthesizing the parallel computational unit for Virtex-6 FPGA.

Table 4.2: Approximated Resource Requirements of Parallel Coefficient Computational Block

Resource	*Number of Units*
Occupied Slices	5995
Slice Registers/FF	778
Slice LUT	21390
IOB	210
DSP48E1	11
% of Slices Used	15%

In comparison to the available resources of the Virtex-6 FPGA, this parallel com-

putational unit can be generated with ample area and resources for implementation on the Virtex-6 FPGA and can be easily scaled up to support higher FIR filter orders.

Tracking the computational performance of this computational unit, the parallel unit was synthesized for the Virtex-6 FPGA at 10.95 MHz reference clock. Furthermore, all 11 of the coefficients are computed and output from the block after 12 cycles. In general, an N-th order filter requires $N + 2$ cycles to recompute the Lagrange coefficients after shifting the delay or changing the order of the filter parameter. As a result, recomputing the coefficients for a 10th order filter takes approximately 1.1 μs. As many FD audio applications sample the input signal on the order of kHz, this recomputation time easily fits within the sampling window. Thus, during input parameter changes, the expectation is that only one sample will drop during the recomputation period, as shown in the software/hardware hybrid filter.

Serial Computational Unit

A serial computational unit was synthesized to compute the same 11 coefficients for a maximum 10th order FIR filter. This block was synthesized and targeted for the Xilinx Spartan-6 FPGA. The primary goal of this design is to create a computational unit that can be implemented on a low-cost FPGA platform. Since the coefficients are computed serially, only one coefficient block is required, as opposed to the need for $N+1$ coefficient blocks for an N-th order filter computed in the parallel computational unit. As a result, the area and resource requirements are substantially less. Table 4.3 lists the approximated resource requirements of synthesizing the serial computational unit for Spartan-6 FPGA. Since each coefficient computation block required a higher

percentage of the overall resources of the Spartan-6 FPGA, only three coefficient computation blocks could be generated for the Spartan-6 FPGA. As a result, the choice was made to target the Spartan-6 FPGA for a serial computation unit.

Table 4.3: Approximated Resource Requirements of Serial Coefficient Computational Block

Resource	*Number of Units*
Occupied Slices	657
Slice Registers/FF	110
Slice LUT	1960
IOB	50
DSP48E1	1
% of Slices Used	26%

The serial unit was synthesized for the Xilinx Spartan-6 FPGA at 6.58 MHz reference clock. To compute all 11 of the coefficients for a 10th order filter, the serial computational unit requires 120 cycles. In general, an N-th order filter requires $N(N+2)$ cycles to recompute the Lagrange coefficients after changing the input parameters. As a result, recomputing the coefficients for the 10th order filter tested takes approximately 18.2 μs. As a result, for a typical audio sampling rate of 44.1 kHz, two samples are dropped during the recomputation time. Thus, there is a performance degradation compared to the performance of the parallel computation unit.

Chapter 5: Conclusion

The novelty of this work is the use of today's (2012) low-cost high performance hardware (FPGA) to compute filter coefficients for varying fractional delay in real time, as the desired delay changes. Traditionally, VFD filter's rely on resource intensive, complex structures to allow levels of varying delay. However, in widely varying fractional delay applications, these complex structures still do not accurately produce fractionally delay signals. Using cost-efficient hardware, a VFD filter was designed that permits widely varying fractional delay at a high precision.

The contribution of this thesis is the development of a prototype of an order-scalable software/hardware hybrid VFD filter targeted for reconfigurable hardware and the design of the hardware components to create a fully hardware-based VFD filter.

The fundamental goal of designing a VFD filter is to design a filter that can reconstruct an interpolated continuous input signal and delay the reconstructed signal by a desired fractional delay. A VFD filter is used in many modern digital signal processing applications, such as echo cancellation, modeling human voice pitch, musical signal analysis, and timing synchronization.

Using Xilinx FPGAs with the Simulink/System Generator environment, a software/hardware hybrid prototype of a VFD filter was created and tested. The proposed filter scales in filter order to permit widely varying fractional delay values with

high accuracy. This prototype was compared to a targeted high performance implementation platform using a Xilinx FPGA paired with a hard processing core via a high-speed bus. Building upon the software/hardware hybrid filter, hardware coefficient computational units were developed and synthesized as building blocks to create a fully hardware VFD filter.

As described in Chapter 4, the order-scaling FIR filter and the hardware-based computational units offer viable performance for typical VFD audio applications. Two hardware-based computational units were developed: a parallel coefficient computation unit and serial coefficient computation unit, targeted for the Xilinx Virtex-6 and Spartan-6 FPGAs, respectively. The parallel unit on the Virtex-6 FPGA offers greater application performance, while the serial unit on the Spartan-6 offers vast area and resource savings.

As part of the future work of this project, the software/hardware hybrid filter architecture can be implemented using a Xilinx FPGA with an integrated hard processor or an embedded processor with a high-speed bus. Porting the architecture to the Xilinx Embedded Design Kit (EDK) can provide for a more high performance test environment. Furthermore, a natural extension to this project is to integrate the hardware computation unit with the order-scaling FIR filter in a fully hardware platform. The Xilinx EDK environment can be used for prototyping a fully-hardware VFD filter.

Bibliography

[1] T I Laakso, Audio Signal Processing, and Computer Technology. Principles of fractional delay filters. *Acoustics, Speech, and Signal Processing*, (June):3–6, 2000.

[2] Shailey Minocha, S C Dutta Roy, and Balbir Kumar. A note on the fir approximation of a fractional sample delay. *International Journal of Circuit Theory and Applications*, 21(3):265–274, 1993.

[3] Hon Keung Kwan and A Jiang. FIR, Allpass, and IIR Variable Fractional Delay Digital Filter Design. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 56(9):2064–2074, 2009.

[4] G. Ramirez-Conejo, J. Diaz-Carmona, A. Ramirez-Agundis, A. Padilla-Medina, and J. Delgado-Frias. FPGA Implementation of Adjustable Wideband Fractional Delay FIR Filters. In *2010 International Conference on Reconfigurable Computing and FPGAs*, pages 406–411. IEEE, December 2010.

[5] U Nithirochananont, S Chivapreecha, and K Dejhan. An FPGA-based implementation of variable fractional delay filter. In *Signal Processing Its Applications, 2009. CSPA 2009. 5th International Colloquium on*, pages 104–107, March 2009.

[6] T I Laakso, V Valimaki, M Karjalainen, and U K Laine. Splitting the unit delay [FIR/all pass filters design]. *IEEE Signal Processing Magazine*, 13(1):30–60, 1996.

[7] G. D. Cain, A. Yardim, and P. Henry. Offset windowing for FIR fractional-sample delay. pages 1276–1279, May 1995.

[8] T I Laakso, T Saramaki, and G D Cain. Asymmetric Dolph-Chebyshev, Saramaki, and transitional windows for fractional delay FIR filter design. In *Circuits and Systems, 1995., Proceedings., Proceedings of the 38th Midwest Symposium on*, volume 1, pages 580 –583 vol.1, August 1995.

[9] A Yardim, G D Cain, and P Henry. Optimal two-term offset windowing for fractional delay. *Electronics Letters*, 32(6):526–527, March 1996.

[10] P J Kootsookos and R C Williamson. FIR approximation of fractional sample delay systems. *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, 43(3):269–271, March 1996.

[11] K M Tsui, S C Chan, and H K Kwan. A New Method for Designing Causal Stable IIR Variable Fractional Delay Digital Filters. *Circuits and Systems II: Express Briefs, IEEE Transactions on*, 54(11):999–1003, 2007.

[12] Aimin Jiang and Hon Keung Kwan. Iterative design of IIR variable fractional delay digital filters. In *Electro/Information Technology, 2009. eit '09. IEEE International Conference on*, pages 163–166, June 2009.

[13] Wu-Sheng Lu and Tian-Bo Deng. An improved weighted least-squares design for variable fractional delay FIR filters. *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, 46(8):1035–1040, August 1999.

[14] H Johansson and P Lowenborg. On the design of adjustable fractional delay FIR filters, 2001.

[15] V Valimaki. *Discrete-Time Modeling of Acoustic Tubes Using Fractional Delay Filters*. Doctoral thesis, Helsinki University of Technology, 1995.

[16] C. T. Roberts, R. A. and Mullis. *Digital Signal Processing*. Addison-Wesley, Reading, Massachusets, 1st edition, 1987.

[17] A J Jerri. The Shannon sampling theorem;Its various extensions and applications: A tutorial review. *Proceedings of the IEEE*, 65(11):1565–1596, June 2005.

[18] F J Harris. On the use of windows for harmonic analysis with the discrete Fourier transform. *Proceedings of the IEEE*, 66(1):51–83, 1978.

[19] Douglas W. Harder. The Vandermonde Matrix, 2009.

[20] F. B Hildebrand. *Introduction to Numerical Analysis*. McGraw-Hill, New York, 2nd edition, 1987.

[21] E Hermanowicz. Explicit formulas for weighting coefficients of maximally flat tunable FIR delayers. *Electronics Letters*, 28(20):1936–1937, 1992.